

# OEES API Reference

## v.2.0.3

---

### API overview

OEES provides a web-service API for programmatic interaction. This API, which is also used by the user-facing OEES application, is divided into a number of *services* (each with its own URL), each of which in turn provides one or more *methods*; each method represents an operation that can be performed. The API can be used to provision, edit, and decommission circuits; retrieve configuration and operational information about a circuit or the network in general; and view or modify administrative state.

The following is documentation for OEES-specific interfaces not involving administrator-level tasks. In addition to the services documented here, there are administrative services, which follow the same general conventions as below, and interfaces for use by NSI-based provisioning agents (and formerly OSCARS – there are references to OSCARS below as a result), which follow different conventions.

### General API description

#### How to get help

The following resources have more information:

<https://globalnoc.iu.edu/sdn/oess.html> — the OEES homepage

<http://github.com/GlobalNOC/OEES> — the latest code, as well as development history and release tags

[oess-users@globalnoc.iu.edu](mailto:oess-users@globalnoc.iu.edu) — the mailing list, if you need to discuss something with a human or haven't been able to answer your question with the previous resources

#### Request format

All services may be called with either GET or POST requests. In the case of a GET request, parameters should be encoded as URL arguments. In the case of a POST request, the parameters should be passed in as application/x-www-form-urlencoded data. Which method on the service to run is indicated by the special **method** parameter. In some methods, a parameter may be list-valued; this is represented by multiple instances of the parameter in the URL or POST data.

Example URL for a GET request calling the `get_circuit_history` method:

`https://<hostname>/oess/services/data.cgi?method=get_circuit_history&circuit_id=999`

#### Response format

All methods return JSON-formatted data; the top-level item is an object. Most methods use the following convention (with the method in measurement.cgi being a notable exception):

If an error occurs, the `error` field of the top-level object will be set, and there may be explanatory text in the `error_text` field; if the method's operation was successful, any results will be put in the `results` field.

In the example responses in this document, there are a number of `// C++-style comments` shown; these are not actually part of the returned response, but serve to help explain the meaning of fields in the returned object.

Example of a response for a successful operation:

```
{  
  "results": [ // array of users' information  
    {  
      "email_address": "johndoe@grnoc.iu.edu",  
      "user_id": "5111" // numeric OEES user ID  
    },  
    {  
      "email_address": "janedoe@grnoc.iu.edu",  
      "user_id": "5122"  
    }  
  [...] // more elements of the array elided for conciseness  
]
```

Example of an error response:

```
{  
  "error_text": "get_existing_circuits: required input parameter workgroup_id is missing ",  
  "error": 1,  
  "results": null  
}
```

## OEES concepts and data types

### Users

An OEES *user* is an entity that performs operations in the OEES system. A user has a unique numeric user ID. Zero or more usernames, as used to authenticate to the web server (see “Authentication” below), are mapped to an OEES user; no username may map to more than one OEES user.

### Workgroups

Every OEES user should be a member of at least one *workgroup*; it is possible for a user to be a member of multiple workgroups. Many operations (e.g., creating a circuit) are performed in the context of a single workgroup, so a particular workgroup must be specified for the operation. Once a user has logged in to the OEES application, they are asked to select a workgroup to work under; people using the API will generally need to do something similar, either in a configuration file or as a run-time argument. Each workgroup may own zero or more endpoint interfaces on the network; no interface is owned by more than one workgroup.

A workgroup has a name (a string; unique within an OEES instance) and an ID (a unique integer). The workgroup ID is used in numerous methods and allows OEES to determine which pieces of data the user is allowed to see and which operations the user may perform.

A workgroup also has a *type*, which is one of *normal* (the vast majority of workgroups in a typical OEES instance), *admin* (usually only one workgroup), or *demo* (usually zero or one workgroup). A workgroup’s type is related to the operations that may be performed using that workgroup; for more details, see “Allowed operations” below.

## Nodes

*Nodes* are switching elements on the network. Each has a unique name (a string) and a unique integer ID; one or the other is used to specify a node, depending on the method in question.

A node may be configured to be used with OpenFlow, with MPLS (typically controlled using NETCONF), or both.

## Interfaces

*Interfaces* are network interfaces on nodes. Each has a name (possibly duplicated on other nodes, but unique for its associated node; a string) and an integer ID (unique in the OEES instance). Depending on the method, an interface may be specified by its ID or by a (node name, interface name) pair. A *trunk interface* is an interface that is an endpoint of a link (see next); an *endpoint interface* is an interface that is not a trunk interface, and marks the boundary of the OEES-managed network (typically, an end system or a different network is on the other side of the boundary). An interface may be owned by a (single) workgroup; except for a few exceptional cases, trunk interfaces do not need to be owned by a workgroup. An interface is OpenFlow-based, MPLS-based, or both. If the same (on-node) interface is accessible from both OpenFlow and MPLS, it may show up in OEES as two interfaces, depending on the details of the node's interface naming scheme and OpenFlow/MPLS implementations.

An interface has an associated list of *ACL entries*, which the owning workgroup of an interface use to allow or deny workgroups (including itself!) the right to terminate circuits at a certain range of VLAN tags on the interface. See the add\_acl method of workgroup\_manage.cgi for more details.

## Links

*Links* are connections between nodes; they connect interfaces on different nodes, and are auto-discovered by OEES (though require confirmation by the administrators to be used). Each has a unique name (a string) and a unique integer ID. Links are OpenFlow-based MPLS-based, or both.

## Circuits

*Circuits* are what most users are ultimately concerned with. A circuit consists of a number of endpoints, specified by (interface, VLAN tag) pairs (the special VLAN tag value –1 refers to untagged frames), and some *paths*, each a set of links over which circuit traffic will travel: a primary path (optional for MPLS-based circuits) and an (optional) secondary path for link-level resiliency. (Behind the scenes, MPLS-based circuits also have a tertiary path, which is used if the primary and secondary paths (if specified) are unusable.)

A circuit has a unique name (a string) and a unique integer ID. It can also be assigned an *external identifier*, which is used by OEES to store circuit identifiers used by OSCARS and NSI agents. A circuit is either OpenFlow-based or MPLS-based – never both.

## Numbers

Numbers are sometimes represented in response fields as strings: comments will generally refer to the value in question being numeric.

## Booleans

Boolean values are represented in method parameters and response fields as 0 (false) or 1 (true). In responses, these may be the strings "0" and "1", instead of the numbers 0 and 1.

## Enumerations

Some method parameters and output fields may only have a limited number of values. This is represented in the parameter/field description using set notation. For example, if a parameter may only have the value “a”, “b”, or “lemon”, its listed value type will be {a,b,lemon}.

## MAC addresses

MAC addresses may be specified in the form

01:02:03:04:0A:0b

or in the form

01-02-03-04-0A-0b

where capitalization doesn’t matter, but field separator (:) or (-) must be consistent in a given address.

## Authentication

OESS relies upon the hosting web server for authentication and application-wide authorization (i.e., whether or not someone is allowed to access OEES at all). The default Apache setup protects OEES using HTTP Basic authentication, backed by an .htpasswd file. However, an instance may use a different authentication mechanism, as configured on the web server. You should contact the administrator of the system to determine which type of authentication should be used for programmatic access. For instance, currently the services for the AL2S instance of OEES may be accessed from an endpoint using Shibboleth browser-centric authentication (for human use of the OEES frontend with federated login) or from an endpoint using HTTP Basic authentication (for easy programmatic use).

As mentioned in “Users” above, zero or more web-server-level usernames are mapped onto a single OEES user; the possibility of the same principal needing to use different kinds of authentication in different contexts is a driver of this layer of indirection.

## Allowed operations (authorization)

Whether or not a certain user may perform an operation as a certain workgroup depends on several factors, some of which are user-centric and some of which are more workgroup-centric.

- First off, a user is either *active* or *decommissioned* (*decom*); a decom user is not allowed to use any methods.
- A user is either *normal* (read-write) or *read-only*; subject to other factors not forbidding an operation, a normal user may use both **RO** methods (which alter neither the network itself nor the OEES database) and **RW** methods (most of which do alter the network, the database, or both), while read-only users may only use **RO** methods. Each method described below is marked **RO** or **RW**.
- There is a concept of *admin user*, used below. An admin user is a user that belongs to at least one workgroup of type *admin*.
- Methods fall into four classes regarding how they use workgroups to determine whether an action is allowed (in all cases, assuming none of the above factors forbid the action):
  - Methods in class **V** have no further restrictions.
  - Methods in class **W** require that the user making the request (1) belong in the workgroup indicated by the **workgroup\_id** parameter or (2) be an admin user.

- Methods in class **C**, which relate to editing circuits, require that the user making the request be (1) an admin user or, failing that, (2) a member of the workgroup indicated by the **workgroup\_id** parameter, which in case (2) must also
  - be the owner of the circuit being operated upon
  - be *active* and not *decom* (workgroups, like users, are either active or decom)
  - not be of type *demo*.<sup>1</sup>
- Methods in class **IN**, which relate to interfaces, require that the user making the request is (1) a member of the workgroup that owns the interface being operated on (which, for these methods, is not necessarily the workgroup indicated by the **workgroup\_id** parameter) or (2) an admin user.

Each method described below is marked with its class.

## Network information service (data.cgi)

**Location:** <https://<hostname>/oess/services/data.cgi>

**Description:** Provides information about the OESS instance and the network it manages. All of these methods leave the OESS instance unchanged; indeed, with one exception (*send\_email*), all of these methods have no effect on the world beyond the fetching of data.

### Method: get\_workgroups

*RO* *V*

Returns a list of the workgroups the current user may work under, including IDs, names, and administrative types.

*No parameters.*

#### Example request:

**method:** get\_workgroups

#### Example response:

```
{
  "results": [ // list of workgroups
    {
      "workgroup_id": "11", // numeric ID of workgroup
      "name": "GRNOC", // human-visible name of workgroup
      "type": "admin" // type of workgroup: {normal,admin,demo}
    },
    {
      "workgroup_id": "51",
      "name": "Indiana GigaPOP",
      "type": "normal"
    },
    {
      "workgroup_id": "831",
      "name": "CIC-UMich",
    }
  ]
}
```

---

<sup>1</sup> provisioning.cgi's *provision\_circuit* method doesn't quite fit this particular sub-part of the ruleset – when creating a circuit, it requires the group not to be *named* “Demo”, as opposed to not being of type *demo*; this is planned to be changed to stock class **C** rules in a future release of OESS.

```

        "type": "normal"
    },
[...]
]
}

```

### Method: get\_maps

R O W

Returns a JSON object representing the network layout, including the OEES-managed network and any networks discovered from the global OSCARS topology. This information is used by the frontend to draw network diagrams, and includes some information on resource-usage limits.

Parameter	Required?	Value type	Description
workgroup_id	yes	integer	Workgroup to gather data as; available-resource counts will be specific to that workgroup
link_type	no	{openflow,mpls}	If specified, limit the links returned to those of the specified type

### Example request:

```

method: get_maps
workgroup_id: 131

```

### Example response:

```

{
  "results": [ // a list of networks, one of which is the OEES-managed network (meta.local == 1), and the others of which are derived from OSCARS topology data (meta.local == 0)
    {
      "nodes": { // the nodes in a network
        "sdn-sw.phoe.net.internet2.edu": { // name of the node, as field name
          "default_forward": "1", // whether to install an OpenFlow rule forwarding LLDP packets on an admin-configured VLAN tag to the OEES OpenFlow controller
          "short_name": null, // name of the node as used in MPLS link discovery
          "tx_delay_ms": "1", // length of time to pause (in millisconds) between consecutive pushes of OpenFlow rule changes to the node from OEES
          "model": null, // name of equipment model; used in selecting device-specific method of MPLS management
          "sw_version": null, // equipment software version; used in selecting device-specific method of MPLS management
          "default_drop": "1", // whether to install an OpenFlow rule to drop frames not matched by other rules
          "node_id": "6701", // OEES numeric ID of node
          "dpid": "3f2718dd388a318", // OpenFlow DPID of node, in hexadecimal
          "tcp_port": "830", // TCP port used to communicate with node for MPLS-based operation
          "node_long": "-112.047533", // longitude of node, in decimal degrees (east is positive)
          "node_name": "sdn-sw.phoe.net.internet2.edu", // name of the node in OEES (or in the OSCARS topology)
          "vlan_range": "100-2500", // VLAN range (or ranges, comma-separated) used for the parts of OpenFlow circuits on trunk interfaces
          "in_maint": "no", // whether the node is in maintenance mode: {yes,no}
          "barrier_bulk": "1", // whether to send one OpenFlow barrier (per node) per overall circuit operation (true) or to send a barrier after every flow modification (false)
        }
      }
    }
  }
}

```

```

    "node_lat": "33.440476", // latitude of node, in decimal degrees (north is positive)
    "mpls": "1", // whether MPLS-based operation is enabled
    "openflow": "1", // whether OpenFlow-based operation is enabled
    "end_epoch": null, // time at which the current maintenance on the node (if any) is
scheduled to end (if there is such a time), in seconds since the Unix epoch
    "max_static_mac_flows": "128", // maximum number of OpenFlow rules involving MAC
addresses that OESS will install on node
    "mgmt_addr": null, // IP address used to communicate with node for MPLS-based
operation
    "max_flows": "1000", // maximum number of OpenFlow rules OESS allows itself to
install on the node
    "number_available_endpoints": 1, // number of endpoint interfaces on the node
available to this workgroup for circuit endpoints
    "vendor": null // name of equipment vendor; used in selecting device-specific method
of MPLS management
  },
  "rtsw.loса.net.internet2.edu": {
    "default_forward": "1",
    "short_name": null,
    "tx_delay_ms": "1",
    "model": null,
    "sw_version": null,
    "default_drop": "0",
    "node_id": "9091",
    "dpid": "3b89c7b2b1",
    "tcp_port": "830",
    "node_long": "-118.295056",
    "node_name": "rtsw.loса.net.internet2.edu",
    "vlan_range": "100-2500",
    "in_maint": "no",
    "barrier_bulk": "1",
    "node_lat": "33.737916",
    "mpls": "1",
    "openflow": "1",
    "end_epoch": "1469790481",
    "max_static_mac_flows": "128",
    "mgmt_addr": null,
    "max_flows": "4000",
    "number_available_endpoints": 4,
    "vendor": null
  },
  [...]
},
"links": { // the inter-node links in a network
  "sdn-sw.phoe.net.internet2.edu": [ // the list of links with an endpoint on this node
  {
    "link_id": "941", // numeric ID of link
    "to": "rtsw.loса.net.internet2.edu", // name of node on other endpoint of link
    "openflow": "1", // whether this link may be used in OpenFlow-based circuits
    "link_capacity": "10000", // claimed data-rate capacity of the link, in Mbps
    "remote_urn": null, // will always be null
    "link_state": "up", // link state of the endpoint interfaces: {up,down,unknown}
    "mpls": "0", // whether this link may be used in MPLS-based circuits
    "link_name": "I2-LOSA-PHOE-100GE-09190", // name of the link
    "maint_epoch": null // time at which the current maintenance on the link (if any)
is scheduled to end (if there is such a time), in seconds since the Unix epoch
  },

```

```

{
  "link_id": "2991",
  "to": "sdn-sw.tucs.net.internet2.edu",
  "openflow": "1",
  "link_capacity": "10000",
  "remote_urn": null,
  "link_state": "up",
  "mpls": "0",
  "link_name": "I2-PHOE-TUCS-100GE-11990",
  "maint_epoch": null
}
],
"rtsw.losa.net.internet2.edu": [
  { // this link shows up under sdn-sw.phoe as well
    "link_id": "941",
    "to": "sdn-sw.phoe.net.internet2.edu",
    "openflow": "1",
    "remote_urn": null,
    "link_capacity": "10000",
    "link_state": "up",
    "mpls": "0",
    "link_name": "I2-LOSA-PHOE-100GE-09190",
    "maint_epoch": null
},
{
  "link_id": "31",
  "to": "rtsw.sunn.net.internet2.edu",
  "openflow": "1",
  "remote_urn": null,
  "link_capacity": "10000",
  "link_state": "up",
  "mpls": "0",
  "link_name": "I2-LOSA-SUNN-100GE-07755",
  "maint_epoch": null
},
[...]
],
[...]
},
"meta": { // information about the network as a whole
  "network_name": "al2s.net.internet2.edu", // name of the network in the OSCARS
topology
  "network_long": "0", // longitude of the network, in decimal degrees (east is
positive); defaults to 0
  "local": 1, // whether this is the network managed by this instance of OESS (true) or
a network derived from the OSCARS topology (false)
  "network_lat": "0" // latitude of the network, in decimal degrees (north is positive);
defaults to 0
}
},
{ // a network derived from OSCARS topology; less detail is available (for instance, no
links)
  "nodes": {
    "transpac.org-rtr.losa": {
      "node_long": "0",
      "node_name": "transpac.org-rtr.losa",
      "node_lat": "0",
      "node_urn": null
    }
  }
}

```

```

        "number_available_endpoints": 0
    }
},
"meta": {
    "network_name": "transpac.org",
    "network_long": "0",
    "local": 0,
    "network_lat": "0"
}
},
[...]
]
}
```

### Method: get\_nodes

*RO* *V*

Returns the list of active nodes.

Parameter	Required?	Value type	Description
type	no	{openflow,mpls, all}	Whether to return the active nodes for which OpenFlow operation is enabled, the active nodes for which MPLS operation is enabled, or all active nodes; if not specified, defaults to “all”

### Example request:

**method:** get\_nodes

### Example response:

```
{
  "results": [ // list of nodes
    {
      "default_forward": "1", // whether to install an OpenFlow rule forwarding LLDP packets
      // on an admin-configured VLAN tag to the OESS OpenFlow controller
      "loopback_address": null, // Loopback IP address of node; used with MPLS-based operation
      "short_name": null, // name of the node as used in MPLS link discovery
      "operational_state_mpls": "unknown", // whether OESS is connected to the node for MPLS
      // management: {up,down,unknown}
      "tx_delay_ms": "1", // length of time to pause (in millisconds) between consecutive
      // pushes of OpenFlow rule changes to the node from OESS
      "model": null, // name of equipment model; used in selecting device-specific method of
      // MPLS management
      "start_epoch": "1474607095", // time of the latest change in the node's admin
      // configuration, in seconds since the Unix epoch
      "sw_version": null, // equipment software version; used in selecting device-specific
      // method of MPLS management
      "admin_state": "active", // state of the node in OESS: {planned,available,active,
      // maintenance,decom}
      "default_drop": "0", // whether to install an OpenFlow rule to drop frames not matched
      // by other rules
      "vlan_tag_range": "101-2500", // set of VLAN tags that may be used on trunk interfaces
      // for OpenFlow-based circuits
      "node_id": "9151", // numeric ID of node
      "latitude": "33.758537", // latitude of node, in decimal degrees (north is positive)
```

```

    "tcp_port": "830", // TCP port used to communicate with node for MPLS-based operation
    "dpid": "857273106838", // OpenFlow DPID of node, in decimal
    "pending_diff": "0", // whether there are MPLS-related configuration changes awaiting
manual approval by an admin
    "longitude": "-84.38759", // longitude of node, in decimal degrees (east is positive)
    "in_maint": "no", // whether the node is in maintenance mode: {yes,no}
    "name": "rtsw.atla.net.internet2.edu", // name of node
    "mpls": "0", // whether the node is configured to be used with MPLS
    "network_id": "1", // ID of the network the node belongs to; all nodes managed by OESS
will belong to the same network - other networks are only used in inter-domain operations
    "openflow": "1", // whether the node is configured to be used with OpenFlow
    "end_epoch": "-1", // will always be -1
    "max_static_mac_flows": "128", // maximum number of OpenFlow rules involving MAC
addresses that OESS will install on node
    "mgmt_addr": null, // IP address used to communicate with node for MPLS-based operation
    "max_flows": "4000", // maximum total number of OpenFlow rules that OESS will install on
node
    "send_barrier_bulk": "1", // whether to send one OpenFlow barrier (per node) per overall
circuit operation (true) or to send a barrier after every flow modification (false)
    "vendor": null, // name of equipment vendor; used in selecting device-specific method of
MPLS management
    "operational_state": "up" // whether the node is connected over OpenFlow to OESS :{up,
down,unknown}
},
[...]
]
}

```

### Method: get\_node\_interfaces

*RO* *>All*

Returns the list of active interfaces on the given node.

Parameter	Required?	Value type	Description
node	yes	string	Name of the node for which to get information
workgroup_id	no	integer	If specified, limit interfaces returned to ones the workgroup owns or has an ACL rule on
show_down	no	boolean	Whether or not to include interfaces that aren't currently link-up in the list; if not specified, defaults to 0
show_trunk	no	boolean	Whether or not to include trunk interfaces in the list; if not specified, defaults to 0
type	no	{openflow,mpls, all}	Whether to include only interfaces enabled for OpenFlow operation, only interfaces enabled for MPLS operation, or both; if not specified, defaults to "all"

### Example request:

```

method: get_node_interfaces
node: sdn-sw.phoe.net.internet2.edu
show_trunk: 1

```

### Example response:

```
{
  "results": [ // array of the active interfaces on the node
    {
      "workgroup_id": null, // numeric ID of owning workgroup, or null if no owner
      "int_role": "trunk", // {trunk, customer, unknown}
      "status": "up", // link state of the interface: {up, down, unknown}
      "interface_id": "55721", // numeric ID of interface in OESS
      "name": "et-7/0/0.0", // name of interface
      "port_number": "53177", // OpenFlow port number for interface, or null if there isn't
      one
      "description": "BACKBONE: LOSA-PHOE ", // textual description of interface
      "workgroup_name": null, // name of owning workgroup, or null if no owner
      "vlan_tag_range": "-1,1-4089", // VLAN tag range allowed for OpenFlow circuits
      "mpls_vlan_tag_range": null // VLAN tag range allowed for MPLS circuits
    },
    {
      "workgroup_id": "461",
      "int_role": "unknown",
      "status": "up",
      "interface_id": "55741",
      "name": "xe-5/0/0.0",
      "port_number": "51281",
      "description": "pas-tst.phoe, em1",
      "workgroup_name": "Performance Assurance",
      "vlan_tag_range": "-1,1-4089",
      "mpls_vlan_tag_range": null
    },
    [...]
  ]
}
```

### Method: `get_interface`

*RO* *V*

Returns information on a single interface.

Parameter	Required?	Value type	Description
interface_id	yes	integer	ID of the interface to retrieve information on

### Example request:

```
method: get_interface
interface_id: 55741
```

### Example response:

```
{
  "results": {
    "workgroup_id": "461", // numeric ID of owning workgroup, or null if there is no owning
    workgroup
    "node_name": "sdn-sw.phoe.net.internet2.edu", // name of node interface is on
    "name": "xe-5/0/0.0", // name of interface
    "interface_id": "55741", // numeric ID of interface
    "port_number": "51281", // OpenFlow port number of interface, if it has one
    "description": "pas-tst.phoe, em1", // textual description of interface
    "workgroup_name": "Performance Assurance", // name of owning workgroup, if there is one
    "vlan_tag_range": "-1,1-4089", // VLAN tag range allowed for OpenFlow circuit endpoints on
```

```

the interface
  "node_id": "6701", // numeric ID of node interface is on
  "speed": "10000", // claimed speed of interface, in Mbps
  "role": "unknown", // {trunk,customer,unknown}
  "operational_state": "up" // the current link state of the interface on the node: {up,
down,unknown}
}
}

```

### Method: get\_workgroup\_interfaces

*RO W*

Returns a list of the interfaces owned by a workgroup.

Parameter	Required?	Value type	Description
workgroup_id	yes	integer	ID of the workgroup to query

### Example request:

```

method: get_workgroup_interfaces
workgroup_id: 51

```

### Example response:

```
{
  "results": [ // list of interfaces
    { // these fields are the same as in get_all_resources_for_workgroup, except that is_owner
and owning_workgroup are not present
      "interface_name": "et-2/3/0.0",
      "vlan_tag_range": "-1,1-4089",
      "remote_links": [],
      "node_name": "rtsw.chic.net.internet2.edu",
      "node_id": "9071",
      "interface_id": "261",
      "description": "I2-S08251 Indiana Gigapop",
      "operational_state": "up"
    },
    [...]
  ]
}
```

### Method: get\_shortest\_path

*RO V*

Returns the shortest contiguous path between the given nodes.

Parameter	Required?	Value type	Description
type	yes	{openflow,mpls}	The type of circuit to calculate a path for
node	yes	array of string	The list of nodes to calculate the shortest path between; there need to be two <i>or more</i> nodes in the list
link	no	array of string	A list of names of links to avoid using in the computed path

### Example request:

```

method: get_shortest_path
node: rtsw.chic.net.internet2.edu
node: sdn-sw.eqch.net.internet2.edu
node: sdn-sw.star.net.internet2.edu
type: openflow
link: I2-CHIC-EQCH-100GE-69888

```

**Example response:**

```
{
  "results": [ // list of links in the shortest path
    {
      "link": "I2-CHIC-EQCH-100GE-55918" // name of link
    },
    {
      "link": "I2-CHIC-STAR-100GE-07743"
    }
  ]
}
```

**Method:** [get\\_existing\\_circuits](#) **RO W**

Returns the list of active circuits owned by a given workgroup or containing an endpoint on an interface owned by the workgroup.

Parameter	Required?	Value type	Description
workgroup_id	yes	integer	ID of the workgroup
path_node_id	no	array of integer	If included, limit the list of circuits to those that traverse one or more of these nodes
endpoint_node_id	no	array of integer	If included, limit the list of circuits to those that have one of more of these nodes as endpoints

**Example request:**

```

method: get_existing_circuits
workgroup_id: 591

```

**Example response:**

```
{
  "results": [ // list of circuits
    {
      "remote_requester": null, // (string) requester of this circuit; used with interdomain
      "circuits
      "last_modified_by": { // details of the user that last modified the circuit
        "status": "active", // administrative state of the user: {active,decom}
        "auth_id": "10515", // internal numeric ID associated with the username
        "family_name": "Doe", // user's family name
        "email": "johnd@globalnoc.iu.edu", // user's email address
        "is_admin": "0", // unused
        "user_id": "4653", // numeric ID of user
        "given_names": "John", // user's given name
        "type": "normal", // read-write/read-only status of user: {normal,read-only}
      }
    }
  ]
}
```

```

        "auth_name": "johnd" // a username associated with the user
    },
    "external_identifier": null, // the external identifier (if any) associated with this
circuit
    "paths": { // the paths defined for this circuit
        "primary": { // indexed by the path's type: {primary,backup,tertiary}
            "circuit_id": "306292", // numeric ID of circuit
            "path_id": "300922", // numeric ID of this path
            "path_instantiation_id": "1202262", // internal ID for the current state of the path
            "status": 1, // collective status of the links in this path; 0 for down, 1 for up, 2
for unknown
            "start_epoch": "1498600161", // time at which this path was last changed in some
way, in seconds since the Unix epoch
                "path_type": "primary", // the path's type: {primary,backup,tertiary}
                "end_epoch": "-1", // will always be -1
                "path_state": "active", // state of the path in OESS: {active,available,deploying}
                "mpls_path_type": "none", // determines how the path is constructed in MPLS
(relevant to MPLS-based circuits only): {strict,loose,none}
                "links": [ // the list of links (if any) in this path
                    {
                        "link_id": "3831", // numeric ID of link
                        "name": "I2-CHIC-EQCH-100GE-55918" // name of link
                    }
                ]
            },
            "backup": {
                "circuit_id": "306292",
                "path_id": "300932",
                "path_instantiation_id": "1202272",
                "status": 1,
                "start_epoch": "1498600161",
                "path_type": "backup",
                "end_epoch": "-1",
                "path_state": "available",
                "mpls_path_type": "none",
                "links": [
                    {
                        "link_id": "221",
                        "name": "I2-CHIC-STAR-100GE-07743"
                    },
                    {
                        "link_id": "3071",
                        "name": "I2-EQCH-STAR-100GE-09299"
                    }
                ]
            },
            "state": "active", // administrative state of the circuit in OESS: {scheduled,deploying,
active,decom,looped,reserved,provisioned}
            "backup_links": [ // list of the links (if any) in the backup path; same format as the
"links" field
            {
                "ip_z": null,
                "node_z": "rtsw.chic.net.internet2.edu",
                "node_a": "sdn-sw.star.net.internet2.edu",
                "name": "I2-CHIC-STAR-100GE-07743",
                "interface_z": "et-2/1/0.0",

```

```

    "port_no_a": "40929",
    "port_no_z": "43583",
    "ip_a": null,
    "interface_z_id": "281",
    "interface_a": "et-0/0/0.0",
    "interface_a_id": "871"
  },
  {
    "ip_z": null,
    "node_z": "sdn-sw.eqch.net.internet2.edu",
    "node_a": "sdn-sw.star.net.internet2.edu",
    "name": "I2-EQCH-STAR-100GE-09299",
    "interface_z": "et-4/0/0.0",
    "port_no_a": "36975",
    "port_no_z": "32695",
    "ip_a": null,
    "interface_z_id": "64671",
    "interface_a": "et-3/0/0.0",
    "interface_a_id": "45661"
  }
],
  "tertiary_links": [], // list of the links (if any) in the tertiary path; same format as
the "links" field
  "remote_url": null, // URL of creating OSCARS instance or NSI agent; for interdomain
circuits
  "created_on": "06/27/2017 21:44:25", // time at which the circuit was created
  "loop_node": null, // numeric ID of node (if any) at which the circuit is Looped, that
is, at which frames received on an interface are sent back out that same interface instead of
being delivered to other links or endpoints
  "links": [ // list of the links in the circuit's primary path
  {
    "ip_z": null, // IP address of Z endpoint (used with MPLS)
    "node_z": "sdn-sw.eqch.net.internet2.edu", // name of node at Z endpoint
    "node_a": "rtsw.chic.net.internet2.edu", // name of node at A endpoint
    "name": "I2-CHIC-EQCH-100GE-55918", // name of the link
    "interface_z": "et-8/1/0.0", // name of interface at Z endpoint
    "port_no_a": "61586", // OpenFlow port number of interface at A endpoint
    "port_no_z": "47864", // OpenFlow port number of interface at Z endpoint
    "ip_a": null, // IP address of A endpoint (used with MPLS)
    "interface_z_id": "72111", // numeric ID of interface at Z endpoint
    "interface_a": "et-8/3/0.0", // name of interface at A endpoint
    "interface_a_id": "64261" // numeric ID of interface at A endpoint
  }
],
  "circuit_id": "306292", // the numeric ID of this circuit
  "static_mac": "0", // whether this is a static-MAC circuit (frames are routed to a
particular endpoint(s) based on their destination MAC addresses; which endpoint corresponds
with which MAC address(es) is statically configured and not auto-learned)
  "workgroup_id": "591", // numeric ID of the workgroup that owns this circuit
  "name": "GRNOC Server Test Ports-c9a9fc8-5b81-11e7-89d1-3777b00d2fc4", // name of
circuit; auto-set by OESS to a unique value upon circuit creation, and intended to be used and
modified (if desired) by external systems
  "description": "doc test circuit", // textual description of circuit
  "endpoints": [ // list of the circuit's endpoint interfaces
  {
    "local": "1", // whether this endpoint node is on the OESS-managed network
    "node": "sdn-sw.eqch.net.internet2.edu", // name of the endpoint node
  }
]
]
```

```

    "mac_addrs": [], // MAC addresses at this endpoint (for static-MAC circuits)
    "interface_description": "pas-tst.eqch, em2", // textual description of interface
    "port_no": "50989", // OpenFlow port number of endpoint interface
    "node_id": "7761", // numeric ID of endpoint node
    "urn": null, // OSCARS link URN for endpoint interface, if any
    "interface": "xe-5/0/1.0", // name of endpoint interface
    "tag": "555", // VLAN tag of endpoint
    "role": "unknown" // role of endpoint interface: {customer,trunk,unknown}
  },
  {
    "local": "1",
    "node": "rtsw.chic.net.internet2.edu",
    "mac_addrs": [],
    "interface_description": "pas-tst.chic em2",
    "port_no": "37410",
    "node_id": "9071",
    "urn": null,
    "interface": "xe-4/2/1.0",
    "tag": "557",
    "role": "unknown"
  }
],
"workgroup": { // the details of the workgroup that owns this circuit
  "workgroup_id": "591", // workgroup's numeric ID
  "status": "active", // is the workgroup active? {active,decom}
  "name": "GRNOC Server Test Ports", // workgroup's name in OEES
  "max_circuit_endpoints": "8", // maximum number of endpoints a new (or newly-modified)
circuit may possess
  "description": "", // textual description of workgroup
  "max_circuits": "100", // maximum number of circuits the workgroup may own at one time
  "external_id": null, // string identifier available for use by external systems to
associate something with the workgroup (optional)
  "type": "normal", // workgroup type: {normal,demo,admin}
  "max_mac_address_per_end": "4" // maximum number of static MAC addresses this
workgroup may use on a single endpoint node
},
"active_path": "primary", // the path currently being used to forward frames: {primary,
backup,tertiary}
  "bandwidth": "0", // bandwidth nominally reserved for circuit, in Mbps
  "internal_ids": { // VLAN tags used on non-endpoint interfaces in the circuit
(meaningful for OpenFlow circuits only)
    "primary": { // tags used in the primary path
      "rtsw.chic.net.internet2.edu": { // indexed by node name
        "64261": "172" // key is interface ID, value is VLAN tag used on that interface
for traffic for this circuit
      },
      "sdn-sw.eqch.net.internet2.edu": {
        "72111": "172"
      }
    },
    "backup": { // tags used in the backup path
      "rtsw.chic.net.internet2.edu": {
        "281": "155"
      },
      "sdn-sw.star.net.internet2.edu": {
        "871": "170",
        "45661": "165"
      }
    }
  }
}

```

```

        },
        "sdn-sw.eqch.net.internet2.edu": {
            "64671": "181"
        }
    },
    "last_edited": "06/27/2017 21:49:21", // time when the latest change to the circuit
happened
    "user_id": "4651", // numeric ID of the user that made the latest change to the circuit;
1 if the latest change to the circuit was an automatic response to a system event (a link
outage, a maintenance, etc.)
    "restore_to_primary": "2", // if non-zero, if the circuit is using the backup or
tertiary path and the links that make up the primary path come back up, the number of minutes
to wait until switching back to using the primary path (assuming it stays up); if zero, do no
such restoration
    "type": "openflow", // type of the circuit: {openflow,mpls}
    "operational_state": "up", // current status of the circuit: {up,down,unknown}
    "created_by": { // details of the user that first created the circuit; same format as
"last_modified_by" field
        "status": "active",
        "auth_id": "10511",
        "family_name": "Doe",
        "email": "jdoe@globalnoc.iu.edu",
        "is_admin": "0",
        "user_id": "4651",
        "given_names": "Jane",
        "type": "normal",
        "auth_name": "jdoe"
    }
},
[...]
]
}

```

### Method: get\_circuits\_by\_interface\_id

*RO* *>All*

Returns the list of circuits that use an interface.

Parameter	Required?	Value type	Description
interface_id	yes	integer	ID of the interface for which to get circuits

#### Example request:

```

method: get_circuits_by_interface_id
interface_id: 66151

```

#### Example response:

```

{
    "results": [ // list of circuits
    {
        "circuit_id": "75931", // numeric ID of circuit
        "name": "I2-MISS2-SALT-VLAN-13287", // name of circuit; auto-set by OESS to a unique
value upon circuit creation, and intended to be used and modified (if desired) by external
systems
        "description": "PerfAssurance: miss2-xe-5/0/0.0 ( seat ) salt-e15/2 vlan 332" // textual
    }
]
}

```

```

        description of circuit
    },
    {
        "circuit_id": "98671",
        "name": "I2-MISS2-SALT-VLAN-47379",
        "description": "test2"
    },
    [...]
}

```

### Method: get\_circuit\_details RO V

Returns all of the details for a given circuit, specified by its ID.

Parameter	Required?	Value type	Description
circuit_id	yes	integer	ID of the circuit to fetch details for

#### Example request:

```

method: get_circuit_details
circuit_id: 75931

```

#### Example response:

```

{
    "results": { // fields are as in get_existing_circuits
        "remote_requester": null,
        "last_modified_by": {
            "status": "active",
            "auth_id": "341",
            "family_name": "Doe",
            "email": "jaked@globalnoc.iu.edu",
            "is_admin": "0",
            "user_id": "101",
            "given_names": "Jake",
            "type": "normal",
            "auth_name": "jaked"
        },
        "external_identifier": null,
        "paths": {
            "primary": {
                "circuit_id": "75931",
                "path_id": "75081",
                "path_instantiation_id": "845551",
                "status": 1,
                "start_epoch": "1467755733",
                "path_type": "primary",
                "end_epoch": "-1",
                "path_state": "active",
                "mpls_path_type": "none",
                "links": [
                    {
                        "link_id": "631",

```

```

        "name": "I2-SALT-SEAT-100GE-08998"
    },
    {
        "link_id": "3231",
        "name": "I2-MISS2-SEAT-100GE-13237"
    }
]
}
},
"state": "active",
"backup_links": [],
"tertiary_links": [],
"remote_url": null,
"created_on": "10/02/2014 09:28:06",
"loop_node": null,
"links": [
{
    "ip_z": null,
    "node_z": "rtsw.seat.net.internet2.edu",
    "node_a": "rtsw.salt.net.internet2.edu",
    "name": "I2-SALT-SEAT-100GE-08998",
    "interface_z": "et-5/0/0.0",
    "port_no_a": "41190",
    "port_no_z": "57426",
    "ip_a": null,
    "interface_z_id": "45921",
    "interface_a": "et-7/3/0.0",
    "interface_a_id": "45631"
},
{
    "ip_z": null,
    "node_z": "rtsw.seat.net.internet2.edu",
    "node_a": "sdn-sw.miss2.net.internet2.edu",
    "name": "I2-MISS2-SEAT-100GE-13237",
    "interface_z": "et-4/1/0.0",
    "port_no_a": "32695",
    "port_no_z": "55810",
    "ip_a": null,
    "interface_z_id": "46051",
    "interface_a": "et-4/0/0.0",
    "interface_a_id": "66161"
}
],
"circuit_id": "75931",
"static_mac": "0",
"workgroup_id": "461",
"name": "I2-MISS2-SALT-VLAN-13287",
"description": "PerfAssurance: miss2-xe-5/0/0.0 ( seat ) salt-e15/2 vlan 332",
"endpoints": [
{
    "local": "1",
    "node": "sdn-sw.miss2.net.internet2.edu",
    "mac_addrs": [],
    "interface_description": "pas-tst.miss2, em1",
    "port_no": "51281",
    "node_id": "7971",
    "urn": null,

```

```
        "interface": "xe-5/0/0.0",
        "tag": "332",
        "role": "unknown"
    },
    {
        "local": "1",
        "node": "rtsw.salt.net.internet2.edu",
        "mac_addrs": [],
        "interface_description": "pas-tst.salt em1",
        "port_no": "50989",
        "node_id": "9061",
        "urn": "urn:ogf:network:domain=al2s.net.internet2.edu:node=sdn-
sw.salt.net.internet2.edu:port=e15/2:link=*",
        "interface": "xe-5/0/1.0",
        "tag": "332",
        "role": "unknown"
    }
],
"workgroup": {
    "workgroup_id": "461",
    "status": "active",
    "name": "Performance Assurance",
    "max_circuit_endpoints": "8",
    "description": "",
    "max_circuits": "400",
    "external_id": null,
    "type": "normal",
    "max_mac_address_per_end": "4"
},
"active_path": "primary",
"bandwidth": "0",
"internal_ids": {
    "primary": {
        "sdn-sw.miss2.net.internet2.edu": {
            "66161": "108"
        },
        "rtsw.seat.net.internet2.edu": {
            "45921": "105",
            "46051": "102"
        },
        "rtsw.salt.net.internet2.edu": {
            "45631": "102"
        }
    }
},
"last_edited": "07/05/2016 21:55:33",
"user_id": "11",
"restore_to_primary": "0",
"type": "openflow",
"operational_state": "up",
"created_by": {
    "status": "active",
    "auth_id": "4941",
    "family_name": "Assurance",
    "email": "devnull@example.net",
    "is_admin": "0",
    "user_id": "1681",

```

```

        "given_names": "Performance",
        "type": "normal",
        "auth_name": "devnull"
    }
}
}

```

### Method: get\_circuit\_details\_by\_external\_identifier RO ✓

Returns all of the details for a circuit, specified by its external identifier. If more than one circuit has the same external identifier, this will return details for one of them.

Parameter	Required?	Value type	Description
external_identifier	yes	string	External identifier given to a circuit

#### Example request:

```

method: get_circuit_details_by_external_identifier
external_identifier: al2s.net.internet2.edu-60421

```

#### Example response:

```
{
  "results": { // fields are as in get_existing_circuits
    "remote_requester": null,
    "last_modified_by": {
      "status": "active",
      "auth_id": "4431",
      "family_name": "Pseudo-user",
      "email": "oscars@example.globalnoc.iu.edu",
      "is_admin": "0",
      "user_id": "401",
      "given_names": "OSCARS",
      "type": "normal",
      "auth_name": "OSCARS"
    },
    "external_identifier": "al2s.net.internet2.edu-60421",
    "state": "decom",
    "backup_links": [],
    "tertiary_links": [],
    "remote_url": null,
    "created_on": "04/22/2015 16:49:25",
    "loop_node": null,
    "links": [
      {
        "ip_z": null,
        "node_z": "rtsw.chic.net.internet2.edu",
        "node_a": "sdn-sw.star.net.internet2.edu",
        "name": "I2-CHIC-STAR-100GE-07743",
        "interface_z": "et-2/1/0.0",
        "port_no_a": "40929",
        "port_no_z": "43583",
        "ip_a": null,
        "interface_z_id": null,
        "interface_a": "et-0/0/0.0",
        "id": "I2-CHIC-STAR-100GE-07743"
      }
    ]
  }
}
```

```

        "interface_a_id": null
    },
],
"circuit_id": "142291",
"static_mac": "0",
"workgroup_id": "1",
"name": "OSCARS IDC-8897a498-e90f-11e4-932b-73e41ce00fa9",
"description": "Test of IDC provisioning",
"endpoints": [
{
    "local": "1",
    "node": "rtsw.chic.net.internet2.edu",
    "mac_addrs": [],
    "interface_description": "pas-tst.chic em2 ",
    "port_no": "37410",
    "node_id": "9071",
    "urn": null,
    "interface": "xe-4/2/1.0",
    "tag": "224",
    "role": "unknown"
},
{
    "local": "1",
    "node": "sdn-sw.star.net.internet2.edu",
    "mac_addrs": [],
    "interface_description": "I2-S10492 CIC OmniPOP ",
    "port_no": "49028",
    "node_id": "9081",
    "urn": null,
    "interface": "et-1/0/0.0",
    "tag": "3804",
    "role": "unknown"
},
{
    "local": "0",
    "node": "oess.dcn.umnet.umich.edu-f10-dynes.dcn.umnet.umich.edu",
    "mac_addrs": [],
    "interface_description": "Te+0/1",
    "port_no": null,
    "node_id": "8221",
    "urn": "urn:ogf:network:domain=oess.dcn.umnet.umich.edu:node=f10-
dynes.dcn.umnet.umich.edu:port=Te+0/1:link=*",
    "interface": "Te+0/1",
    "tag": "238",
    "role": "unknown"
}
],
"workgroup": {
    "workgroup_id": "1",
    "status": "active",
    "name": "OSCARS IDC",
    "max_circuit_endpoints": "8",
    "description": "",
    "max_circuits": "100",
    "external_id": null,
    "type": "normal",
    "max_mac_address_per_end": "4"
}
]
```

```

        },
        "active_path": "primary",
        "bandwidth": "5",
        "internal_ids": {},
        "last_edited": "04/22/2015 17:03:03",
        "user_id": "401",
        "restore_to_primary": "0",
        "type": "openflow",
        "operational_state": "unknown",
        "created_by": {
            "status": "active",
            "auth_id": "4431",
            "family_name": "Pseudo-user",
            "email": "oscars@example.globalnoc.iu.edu",
            "is_admin": "0",
            "user_id": "401",
            "given_names": "OSCARS",
            "type": "normal",
            "auth_name": "OSCARS"
        }
    }
}

```

### Method: get\_circuit\_scheduled\_events RO

Returns the list of actions scheduled for a circuit.

Parameter	Required?	Value type	Description
circuit_id	yes	integer	ID of the circuit to get actions for

#### Example request:

```

method: get_circuit_scheduled_events
circuit_id: 306302

```

#### Example response:

```

{
  "results": [
    {
      "activated": "2017-06-29 00:00:00", // when the action is scheduled to be performed
      "scheduled_action_id": "1962752", // numeric ID of scheduled action
      "scheduled": "2017-06-28 17:00:57", // when the action was created
      "username": "zcatlin", // a username associated with the user that scheduled the action
      "layout": "<opt name=\"\" action=\"remove\" version=\"1.0\" />\n", // XML description of
      scheduled action
      "fullname": "Zachary Catlin", // name of user that scheduled the action
      "user_id": "1729", // numeric ID of user that scheduled the action
      "completed": null // when the action was completed; will always be null here
    }
  ]
}

```

### Method: get\_circuit\_history RO

Returns the list of events (both user-initiated and network-driven) that have affected a circuit.

Parameter	Required?	Value type	Description
circuit_id	yes	integer	ID of the circuit to get actions for

**Example request:**

```
method: get_circuit_history
circuit_id: 306302
```

**Example response:**

```
{
  "results": [ // list of events
    {
      "ended": null, // when the next event occurred, if any has
      "layout": "", // always the empty string
      "activated": "2017-06-28 17:00:13", // when the event occurred
      "reason": "User requested circuit edit", // brief textual description of why the event
      occurred
      "fullname": "Zachary Catlin", // name of user that performed this action (if user-
      initiated)
      "scheduled": -1, // always -1
      "username": "zcatlin" // a username associated with the user that performed the action
      (if user-initiated)
    },
    {
      "ended": "2017-06-28 17:00:13",
      "layout": "",
      "activated": "2017-06-28 16:57:10",
      "reason": "CHANGE PATH: User requested",
      "fullname": "Jane Doe",
      "scheduled": -1,
      "username": "jdoe"
    },
    {
      "ended": "2017-06-28 16:57:10",
      "layout": "",
      "activated": "2017-06-28 16:56:39",
      "reason": "Circuit Creation",
      "fullname": "Zachary Catlin",
      "scheduled": -1,
      "username": "zcatlin"
    }
  ]
}
```

**Method: is\_vlan\_tag\_available**

*RO* *V*

Returns whether a given VLAN tag on a given interface is available for new use; also returns the type of circuit that can use that tag.

Parameter	Required?	Value type	Description
node	yes	string	Name of the node
interface	yes	string	Name of the interface on the node
vlan	yes	integer	VLAN tag on the interface to check availability on

<code>workgroup_id</code>	no	integer	Workgroup for which to see whether the VLAN tag is available; if this isn't specified, an affirmative result is returned only if the VLAN tag is available for <i>all</i> workgroups
---------------------------	----	---------	--

**Example request:**

```
method: is_vlan_tag_available
node: rtsw.seat.net.internet2.edu
interface: xe-8/0/1.0
vlan: 206
workgroup_id: 5992
```

**Example response:**

```
{
  "results": [
    {
      "type": "openflow", // type of circuit that may use this VLAN tag: {openflow,mpls}; not always present if available == 0
      "available": 1 // whether the VLAN tag may be available for use
    }
  ]
}
```

**Method: get\_workgroup\_members** **RO W**

Returns the list of users in a workgroup.

Parameter	Required?	Value type	Description
<code>workgroup_id</code>	yes	integer	ID of the workgroup to get members for
<code>order_by</code>	no	{given_names, auth_name}	Order the users by their given <sup>2</sup> names (the default) or by their usernames

**Example request:**

```
method: get_workgroup_members
workgroup_id: 99
```

**Example response:**

```
{
  "results": [ // the list of OESS users that belong to the workgroup
    {
      "email_address": "jdoe@globalnoc.iu.edu", // user's email address
      "status": "active", // user's status in OESS: {active,decom}
      "user_id": "341", // user's numeric ID
      "family_name": "Doe", // user's family name
      "auth_name": [ // web-server-level username(s) for the user
        "jdoe"
      ]
    }
  ]
}
```

---

<sup>2</sup> Given, as opposed to family, name. In the Western convention for names, this is the first name.

```

        ],
        "first_name": "Jane" // user's given name
    },
    {
        "email_address": "zcatlin@example.edu",
        "status": "active",
        "user_id": "1729",
        "family_name": "Catlin",
        "auth_name": [
            "zcatlin",
            "zcatlin2"
        ],
        "first_name": "Zachary"
    },
    [...]
}

```

### Method: generate\_clr

*RO* *V*

Returns a human-readable circuit layout record (CLR) describing the given circuit.

Parameter	Required?	Value type	Description
circuit_id	yes	integer	ID of the circuit to generate a CLR for
raw	no	boolean	If set to 1, instead of the normal CLR view, return a textual description of the OpenFlow rules used to create the circuit (if the circuit is OpenFlow-based). Defaults to 0.

#### Example request:

```

method: generate_clr
circuit_id: 306302

```

#### Example response:

```

{
    "results": {
        "clr": "Circuit: GRNOC Server Test Ports-c14c9c3e-5c22-11e7-89d1-3777b00d2fc4\nCreated by: Zachary Catlin at 06/28/2017 16:56:39 for workgroup GRNOC Server Test Ports\nLast Modified By: Zachary Catlin at 06/28/2017 17:00:13\n\nEndpoints: \n    sdn-sw.miss2.net.internet2.edu - xe-5/0/0.0 VLAN 406\n    rtsw.seat.net.internet2.edu - xe-8/0/1.0 VLAN 360\n\nActive Path:\n    primary\n    Primary Path:\n        I2-MISS2-SEAT-100GE-13237\n    Backup Path:\n        I2-DENV-KANS-100GE-07746\n        I2-DENV-SALT-100GE-07747\n        I2-SALT-SEAT-100GE-08998\n        I2-MINN-MISS2-100GE-09130\n    I2-KANS-MINN-100GE-12675\n" // textual CLR
    }
}

```

#### Example request:

```

method: generate_clr
circuit_id: 306302
raw: 1

```

**Example response:**

```
{  
    "results": {  
        "clr": "OFFlowMod:\n DPID: 64649b693cc0 (sdn-sw.miss2.net.internet2.edu)\n Priority:  
32768\n Match: VLAN: 154, IN PORT: 32695 (et-4/0/0.0)\n Actions: SET VLAN ID: 406\n  
OUTPUT: 51281 (xe-5/0/0.0) \n\nOFFlowMod:\n DPID: 204e71ce76c0 (rtsw.seat.net.internet2.edu)\n  
Priority: 32768\n Match: VLAN: 152, IN PORT: 55810 (et-4/1/0.0)\n Actions: SET VLAN ID: 360\n  
OUTPUT: 47282 (xe-8/0/1.0) \n\nOFFlowMod:\n DPID: ac4bc84207c0 (sdn-  
sw.denv.net.internet2.edu)\n Priority: 32768\n Match: VLAN: 255, IN PORT: 40909 (et-8/0/0.0)\n  
Actions: SET VLAN ID: 185\n OUTPUT: 32695 (et-4/0/0.0) \n\nOFFlowMod:\n DPID:  
ac4bc84207c0 (sdn-sw.denv.net.internet2.edu)\n Priority: 32768\n Match: VLAN: 189, IN PORT:  
32695 (et-4/0/0.0)\n Actions: SET VLAN ID: 239\n OUTPUT: 40909 (et-8/0/0.0)  
\n\nOFFlowMod:\n DPID: d404ff6c61c0 (rtsw.kans.net.internet2.edu)\n Priority: 32768\n Match:  
VLAN: 239, IN PORT: 55810 (et-4/1/0.0)\n Actions: SET VLAN ID: 156\n OUTPUT: 47864  
(et-8/1/0.0) \n\nOFFlowMod:\n DPID: d404ff6c61c0 (rtsw.kans.net.internet2.edu)\n Priority:  
32768\n Match: VLAN: 156, IN PORT: 47864 (et-8/1/0.0)\n Actions: SET VLAN ID: 255\n  
OUTPUT: 55810 (et-4/1/0.0) \n\nOFFlowMod:\n DPID: ac4bc88ebfc0 (sdn-  
sw.minn.net.internet2.edu)\n Priority: 32768\n Match: VLAN: 152, IN PORT: 32695 (et-4/0/0.0)\n  
Actions: SET VLAN ID: 156\n OUTPUT: 40909 (et-8/0/0.0) \n\nOFFlowMod:\n DPID:  
ac4bc88ebfc0 (sdn-sw.minn.net.internet2.edu)\n Priority: 32768\n Match: VLAN: 156, IN PORT:  
40909 (et-8/0/0.0)\n Actions: SET VLAN ID: 156\n OUTPUT: 32695 (et-4/0/0.0)  
\n\nOFFlowMod:\n DPID: 307c5e94e440 (rtsw.salt.net.internet2.edu)\n Priority: 32768\n Match:  
VLAN: 185, IN PORT: 55810 (et-4/1/0.0)\n Actions: SET VLAN ID: 133\n OUTPUT: 41190  
(et-7/3/0.0) \n\nOFFlowMod:\n DPID: 307c5e94e440 (rtsw.salt.net.internet2.edu)\n Priority:  
32768\n Match: VLAN: 130, IN PORT: 41190 (et-7/3/0.0)\n Actions: SET VLAN ID: 189\n  
OUTPUT: 55810 (et-4/1/0.0) \n\nOFFlowMod:\n DPID: 64649b693cc0 (sdn-  
sw.miss2.net.internet2.edu)\n Priority: 32768\n Match: VLAN: 156, IN PORT: 53177 (et-  
7/0/0.0)\n Actions: SET VLAN ID: 406\n OUTPUT: 51281 (xe-5/0/0.0) \n\nOFFlowMod:\n DPID:  
204e71ce76c0 (rtsw.seat.net.internet2.edu)\n Priority: 32768\n Match: VLAN: 133, IN  
PORT: 57426 (et-5/0/0.0)\n Actions: SET VLAN ID: 360\n OUTPUT: 47282 (xe-8/0/1.0)  
\n\nOFFlowMod:\n DPID: 64649b693cc0 (sdn-sw.miss2.net.internet2.edu)\n Priority: 32768\n  
Match: VLAN: 406, IN PORT: 51281 (xe-5/0/0.0)\n Actions: SET VLAN ID: 152\n OUTPUT:  
32695 (et-4/0/0.0) \n\nOFFlowMod:\n DPID: 204e71ce76c0 (rtsw.seat.net.internet2.edu)\n  
Priority: 32768\n Match: VLAN: 360, IN PORT: 47282 (xe-8/0/1.0)\n Actions: SET VLAN ID: 154  
OUTPUT: 55810 (et-4/1/0.0) \n\n// textual description of OpenFlow rules used in circuit  
    }  
}
```

**Method: get\_all\_node\_status**RO 

Returns a list of all active nodes and their current operational status.

Parameter	Required?	Value type	Description
type	no	{openflow,mpls, all}	Whether to return the active nodes for which OpenFlow operation is enabled, the active nodes for which MPLS operation is enabled, or all active nodes; if not specified, defaults to “all”

**Example request:**

```
method: get_all_node_status
```

**Example response:**

```

{
  "results": [ // list of nodes
    { // fields for a node are just like in the get_nodes method
      "default_forward": "1",
      "loopback_address": null,
      "short_name": null,
      "operational_state_mpls": "unknown",
      "tx_delay_ms": "1",
      "model": null,
      "start_epoch": "1474607095",
      "sw_version": null,
      "admin_state": "active",
      "default_drop": "0",
      "vlan_tag_range": "101-2500",
      "node_id": "9151",
      "latitude": "33.758537",
      "tcp_port": "830",
      "dpid": "3087146175882",
      "pending_diff": "0",
      "longitude": "-84.38759",
      "in_maint": "no",
      "name": "rtsw.atla.net.internet2.edu",
      "mpls": "1",
      "network_id": "1",
      "openflow": "1",
      "end_epoch": "-1",
      "max_static_mac_flows": "128",
      "mgmt_addr": null,
      "max_flows": "4000",
      "send_barrier_bulk": "1",
      "vendor": null,
      "operational_state": "up"
    },
    {
      "default_forward": "1",
      "loopback_address": null,
      "short_name": null,
      "operational_state_mpls": "unknown",
      "tx_delay_ms": "1",
      "model": null,
      "start_epoch": "1468565197",
      "sw_version": null,
      "admin_state": "active",
      "default_drop": "0",
      "vlan_tag_range": "100-2500",
      "node_id": "9071",
      "latitude": "41.896504",
      "tcp_port": "830",
      "dpid": "46165784665004",
      "pending_diff": "0",
      "longitude": "-87.64306",
      "in_maint": "no",
      "name": "rtsw.chic.net.internet2.edu",
      "mpls": "1",
      "network_id": "1",
      "openflow": "1",
      "end_epoch": "-1",
    }
  ]
}

```

```

        "max_static_mac_flows": "128",
        "mgmt_addr": null,
        "max_flows": "4000",
        "send_barrier_bulk": "1",
        "vendor": null,
        "operational_state": "up"
    },
    [...]
]
}

```

### Method: get\_all\_link\_status

*RO* *V*

Returns a list of all active links usable by OpenFlow-based circuits and their current operational status.

Parameter	Required?	Value type	Description
type	no	{openflow,mpls,all}	Whether to return the active links usable by OpenFlow-based circuits, the active links usable by MPLS-based circuits, or both, respectively; if not specified, defaults to “all”

#### Example request:

**method:** get\_all\_link\_status

#### Example response:

```
{
  "results": [
    {
      "fv_status": "up", // whether the link is up, as determined by the Flow Verification
      daemon: {up,down,unknown}
      "ip_z": null, // Z endpoint's IP address (used for MPLS only)
      "in_maint": "no", // is the link currently in maintenance mode? {yes,no}
      "status": "up", // whether the link's endpoint interfaces both indicated the link is up:
      {up,down,unknown}
      "remote_urn": null, // will always be null
      "start_epoch": "1373693892", // time (in seconds since the Unix epoch) when the current
      operational status of the link began
      "metric": "224", // numeric measure of cost of using this link in a circuit (e.g.,
      length of link, or length/bandwidth, etc.); used by the get_shortest_path method
      "name": "I2-ALBA-BOST-100GE-09210", // name of the link
      "mpls": "0", // whether the link may be used in MPLS-based circuits
      "link_id": "961", // numeric ID of the link
      "openflow": "1", // whether the link may be used in OpenFlow-based circuits
      "vlan_tag_range": null, // VLAN tag range used with OpenFlow-based circuits using this
      link; null means to use the trunk tag range of the link's endpoint nodes
      "end_epoch": "-1", // time (in seconds since the Unix epoch) when the current
      operational status of the link ended; will always be -1, which means “status is still current”
      "ip_a": null, // A endpoint's IP address (used for MPLS only)
      "interface_z_id": "55761", // Z endpoint's interface ID
      "link_state": "active", // Administrative state of the link in OESS: {planned,available,
      active,maintenance,decom}
      "interface_a_id": "54401" // A endpoint's interface ID
    },
  ]
}
```

```
{
  "fv_status": "up",
  "ip_z": null,
  "in_maint": "no",
  "status": "up",
  "remote_urn": null,
  "start_epoch": "1373693892",
  "metric": "760",
  "name": "I2-ALBA-CLEV-100GE-07735",
  "mpls": "0",
  "link_id": "971",
  "openflow": "1",
  "vlan_tag_range": null,
  "end_epoch": "-1",
  "ip_a": null,
  "interface_z_id": "55771",
  "link_state": "active",
  "interface_a_id": "31"
},
[...]
}
```

### Method: get\_users

**RO ADMIN**

Returns a list of all OESS users.

Parameter	Required?	Value type	Description
-----------	-----------	------------	-------------

#### Example request:

**method:** get\_users

#### Example response:

```
{
  "results": [ // list of users
  {
    "email_address": "user@demo.edu",
    "status": "active",
    "type": "normal",
    "user_id": "7",
    "family_name": "User",
    "auth_name": ["demo"],
    "first_name": "Demo",
  },
  ...
]
```

## Method: get\_all\_resources\_for\_workgroup

RO W

Returns the list of all interfaces the workgroup either owns or can otherwise provision circuits on, included allowed VLAN tag ranges.

Parameter	Required?	Value type	Description
workgroup_id	yes	integer	ID of the workgroup to get the resource list for

### Example request:

```
method: get_all_resources_for_workgroup
workgroup_id: 461
```

### Example response:

```
{
  "results": [ // list of interfaces
    {
      "interface_name": "et-4/3/0.0", // name of this interface
      "remote_links": [], // a set of (remote_urn, vlan_tag_range)s for inter-domain links
      associated with this interface
      "node_name": "rtsw.atla.net.internet2.edu", // name of the node this interface is on
      "owning_workgroup": { // the details of the workgroup that owns this interface; not
      always present if the same as workgroup_id
        "workgroup_id": "521", // workgroup's numeric ID
        "status": "active", // is the workgroup active? {active,decom}
        "name": "SoX", // workgroup's name in OEES
        "max_circuit_endpoints": "8", // maximum number of endpoints a new (or newly-modified)
        circuit may possess
        "description": "", // textual description of workgroup
        "max_circuits": "100", // maximum number of circuits the workgroup may own at one time
        "external_id": "I2-S10678", // string identifier available for use by external systems
        to associate something with the workgroup (optional)
        "type": "normal", // workgroup type: {normal,demo,admin}
        "max_mac_address_per_end": "4" // maximum number of static MAC addresses this
        workgroup may use on a single endpoint node
      },
      "interface_id": "53931", // numeric ID of interface
      "description": "I2-S11441 SOX ", // textual description of interface
      "is_owner": 0, // whether workgroup_id is the owner of this interface
      "vlan_tag_range": "1650", // set of VLAN tags workgroup_id may use as an endpoint on
      this interface
      "node_id": "9151", // numeric ID of the node this interface is on
      "operational_state": "up" // link-up state of this interface: {up,down,unknown}
    },
    [...]
  ]
}
```

## Method: send\_email

RO V

Sends a message to the instance's admin email address.

Parameter	Required?	Value type	Description
subject	yes	string	Subject of email
body	yes	string	Body of email

**Example request:**

```
method: send_email
subject: Test Email
body: testing 1 2 3
```

**Example response:**

```
{
  "results": [
    {
      "success": 1 // [sic]; whether the email was successfully sent
    }
  ]
}
```

**Method:** get\_link\_by\_name RO  $\forall$

Returns information about a link given the link's name.

Parameter	Required?	Value type	Description
name	yes	string	Name of the link

**Example request:**

```
method: get_link_by_name
name: I2-ALBA-BOST-100GE-09210
```

**Example response:**

```
{
  "results": { // the following fields are a subset of those in the response from
get_all_link_status
  "fv_status": "up",
  "in_maint": "no",
  "status": "up",
  "remote_urn": null, // OSCARS URN of link; will be null for links in the OESS-managed
network
  "metric": "224",
  "name": "I2-ALBA-BOST-100GE-09210",
  "link_id": "961",
  "vlan_tag_range": null
}
}
```

**Method:** is\_within\_mac\_limit RO  $\forall$

Returns whether the specified MAC address(es) may all be added to an interface without going over workgroup, node, etc. MAC-address limits.

Parameter	Required?	Value type	Description
workgroup_id	yes	integer	ID of workgroup to evaluate under
node	yes	string	Name of node
interface	yes	string	Name of interface on node
mac_address	yes	array of MAC address	List of MAC addresses to be added to a circuit termination on that interface

**Example request:**

```

method: is_within_mac_limit
workgroup_id: 461
node: rtsw.seat.net.internet2.edu
interface: xe-8/0/1.0
mac_address: 01-02-03-04-05-06
mac_address: 07:08:09:10:11:12

```

**Example response:**

```
{
  "error": null,
  "results": [
    {
      "explanation": null, // if verified == 0, a textual explanation of why the MAC
      address(es) may not all be added
      "verified": 1 // whether the MAC address(es) may all be added
    }
  ]
}
```

**Method: is\_within\_circuit\_endpoint\_limit**

*RO* *V*

Returns whether or not a workgroup may create a circuit with the specified number of endpoints.

Parameter	Required?	Value type	Description
workgroup_id	yes	integer	ID of workgroup in question
endpoint_num	yes	integer	Number of endpoints in proposed circuit

**Example request:**

```

method: is_within_circuit_endpoint_limit
workgroup_id: 461
endpoint_num: 23

```

**Example response:**

```
{
  "error": null,
  "results": [
    {
      "within_limit": 0 // whether the workgroup may create a circuit with endpoint_num
      endpoints
    }
  ]
}
```

```

        }
    ]
}
```

### Method: is\_within\_circuit\_limit

*RO* *V*

Returns whether or not a workgroup may create more circuits. The method name is slightly misleading: for example, if a workgroup is set to have a limit of 10 circuits and currently has 10 circuits, `within_limit` will be 0, as no more circuits may be created by the workgroup, even though it is still within the limit.

Parameter	Required?	Value type	Description
<code>workgroup_id</code>	yes	integer	ID of workgroup to query

### Example request:

```

method: is_within_circuit_limit
workgroup_id: 461
```

### Example response:

```
{
  "error": null,
  "results": [
    {
      "within_limit": 1 // whether a new circuit can be created
    }
  ]
}
```

### Method: get\_vlan\_tag\_range

*RO* *V*

For the given interface, returns the set of VLAN tags the specified workgroup may use when creating circuits as a comma-delimited list of ranges.

Parameter	Required?	Value type	Description
<code>workgroup_id</code>	yes	integer	ID of workgroup to query for
<code>node</code>	yes	string	Name of node
<code>interface</code>	yes	string	Name of interface on node

### Example request:

```

method: get_vlan_tag_range
workgroup_id: 92
node: rtsw.salt.net.internet2.edu
interface: et-4/3/0.0
```

### Example response:

```
{
  "results": [
    {
      "vlan_tag_range": "500,2000-2099,3551-3600" // set of allowed VLAN tags
    }
  ]
}
```

```

        }
    ]
}

```

## Provisioning service (provisioning.cgi)

**Location:** <https://<hostname>/oess/services/provisioning.cgi>

**Description:** Provides methods to add, modify, or remove circuits to/from the network.

**Method:** provision\_circuit

**RW C**

Adds or modifies a circuit on the network. If circuit\_id is undefined or –1, then the circuit is added; otherwise, the circuit with the specified ID is modified.

Parameter	Required?	Value type	Description
circuit_id	no	integer	The ID of the circuit to modify, if specified and not –1; otherwise, a new circuit is (attempted to be) created
workgroup_id	yes	integer	ID of the workgroup to own the circuit (if creating) or which owns the circuit (if modifying)
provision_time	yes	integer	The time at which to add/modify the circuit, in seconds since the Unix epoch; the special value –1 means “now”
remove_time	yes	integer	The time at which to automatically remove the circuit, in seconds since the Unix epoch; the special value –1 means “never”
external_identifier	no	string	If specified at circuit creation, the external identifier to associate with this circuit
description	yes	string	A short description of the circuit; free-form text
bandwidth	no	integer	Reserved bandwidth for circuit, in Mbps; not currently used
restore_to_primary	no	integer	If specified and non-zero, the number of minutes to wait after the primary path becomes usable before automatically switching the circuit to using the primary path again; if not specified or zero, don't automatically restore to primary path
static_mac	no	boolean	Whether the circuit uses static MAC-address routing; defaults to 0
link	no	array of string	The names of the links to use for the circuit's primary path  (required if circuit is OpenFlow-based)

backup_link	no	array of string	The names of the links to use for the circuit's backup path, if one is desired
node	no	array of string	The <i>n</i> th entry is the name of the node for the circuit's <i>n</i> th endpoint
interface	no	array of string	The <i>n</i> th entry is the name of the interface for the circuit's <i>n</i> th endpoint
tag	no	array of integer	The <i>n</i> th entry is the VLAN tag for the circuit's <i>n</i> th endpoint
inner_tag	no	array of integer	The <i>n</i> th entry is the Inner VLAN tag for the circuit's <i>n</i> th endpoint. If specified tag becomes the S-Tag of a QnQ tagged interface.
endpoint_mac_address_num	no	array of integer	Used with static-MAC circuits; the <i>n</i> th entry is the number of MAC addresses for the <i>n</i> th endpoint
mac_address	no	array of MAC address	Used with static-MAC circuits; the list of MAC addresses to associate with endpoints; they should be arranged in the same endpoint order as node, interface, etc., but multiple MAC addresses may be associated with an endpoint, as controlled by endpoint_mac_address_num
loop_node	no	string	Specified and used only when doing loopback testing of an existing circuit; name of the node at which to hairpin frames back to their source interface
state	no	string	State of the circuit; defaults to "active"
remote_node	no	array of string	Array of OSCARS URNs to use as endpoints for IDC-based circuits
remote_tag	no	array of integer	VLAN tags to be used on IDC endpoints; the <i>m</i> th entry will be used for the <i>m</i> th entry of remote_node
remote_url	no	string	URL of creating OSCARS instance or NSI agent, for interdomain circuits
remote_requester	no	string	Requester of an interdomain circuit
endpoint	no	JSON	JSON object describing desired Endpoint

**Example request:**

```

method: provision_circuit
workgroup_id: 491
provision_time: -1
remove_time: 1500100100
description: API Test Circuit

```

```

static_mac: 0
link: I2-CHIC-EQCH-100GE-55918
backup_link: I2-EQCH-STAR-100GE-09299
backup_link: I2-CHIC-STAR-100GE-07743
endpoint: {
    "entity": "Demo Network", // Network Entity to connect to
    "tag": 300, // VLAN to use for connection
    "bandwidth": 0, // Max Bandwidth on Endpoint. 0 is Unlimited
}

```

**Example response:**

```
{
  "results": {
    "circuit_id": "306322", // numeric ID of the circuit
    "success": 1 // whether the circuit was successfully created/modified
  }
}
```

**Method:** fail\_over\_circuit RWC

If a circuit is using its primary path, change it to use its backup path (if it has one); if the circuit is using its backup path, change it to use its primary path.

Parameter	Required?	Value type	Description
circuit_id	yes	integer	ID of the circuit to change active path on
workgroup_id	yes	integer	ID of the workgroup to perform this action as
force	no	boolean	Whether or not to change paths even if the alternate path isn't currently up; defaults to 0

**Example request:**

```

method: fail_over_circuit
circuit_id: 306322
workgroup_id: 491

```

**Example response:**

```
{
  "results": [
    {
      "success": 1 // whether the circuit's path was actually changed
    }
  ]
}
```

**Method:** reprovision\_circuit RWC

Removes and re-installs the circuit on the network; can be useful for troubleshooting.

Parameter	Required?	Value type	Description
circuit_id	yes	integer	ID of circuit to re-provision

<code>workgroup_id</code>	yes	integer	ID of workgroup to run this as
<code>type</code>	no	{openflow,mpls}	Unused

**Example request:**

```
method: reprovision_circuit
circuit_id: 306322
workgroup_id: 491
```

**Example response:**

```
{
  "results": [
    {
      "success": 1 // whether the circuit was successfully reprovisioned
    }
  ]
}
```

**Method: remove\_circuit** RWC

Removes a circuit from the network (or schedules such removal), and returns success if the circuit has been removed successfully or is scheduled for removal from the network.

Parameter	Required?	Value type	Description
<code>circuit_id</code>	yes	integer	ID of the circuit to remove
<code>workgroup_id</code>	yes	integer	ID of the workgroup to perform this operation as
<code>type</code>	no	{openflow,mpls}	Unused
<code>remove_time</code>	yes	integer	Time to remove the circuit, in seconds since the Unix epoch; -1 means “now”
<code>force</code>	no	boolean	Whether or not to decommission the circuit in OESS’s database in the case that removing the circuit’s configuration from the nodes fails for some reason

**Example request:**

```
method: remove_circuit
circuit_id: 306322
workgroup_id: 491
remove_time: -1
```

**Example response:**

```
{
  "results": [
    {
      "success": 1 // whether the circuit was successfully removed
    }
  ]
}
```

## Measurement service (measurement.cgi)

**Location:** <https://<hostname>/oess/services/measurement.cgi>

**Description:** Provides network usage data about circuits.

Method: get\_circuit\_data

RO  $\forall$

Returns network usage data relating to a given circuit on a given interface. **NOTE:** the returned data does not follow the usual JSON format.

Parameter	Required?	Value type	Description
circuit_id	yes	integer	ID of the circuit to get usage data for
node	no	string	Name of the node; if neither this nor link is specified, a node is chosen at random
interface	no	string	Name of the interface on the node to show data for; if not specified, an interface is chosen at random
link	no	string	Name of a link involved in the circuit; if specified, node and interface are ignored, and data will be shown for one of the link's endpoint interfaces
start	yes	unsigned integer	Start time of data to retrieve, in seconds since the Unix epoch
end	yes	unsigned integer	End time of data to retrieve, in seconds since the Unix epoch

**Example request:**

```
method: get_circuit_data
circuit_id: 43224
node: sdn-sw.denv.net.internet2.edu
interface: eth5/2
start: 1498679067
end: 1498679667
```

**Example response:**

```
{
  "interfaces": [ // names of the interfaces on the same node that are involved in the circuit
    "eth5/2",
    "eth3/5"
  ],
  "interface": "eth5/2", // name of the interface the usage data is for
  "node": "sdn-sw.denv.net.internet2.edu", // name of the node the usage data is for
  "results": [ // array of different data values
    {
      "name": "Input (Bps)", // name of the data value in question
      "data": [ // array of (timestamp, data at timestamp) pairs
        [ 1498679160, 207157.893736168 ],
        [ 1498679280, 56469466.0722517 ],
        ...
      ]
    }
  ]
}
```

```

        [ 1498679400, 59934688.3466288 ],
        [ 1498679520, 241695.117964083 ],
        [ 1498679640, 227999.937238485 ]
    ]
},
{
  "name": "Output (Bps)",
  "data": [
    [ 1498679160, 195704.405191311 ],
    [ 1498679280, 3377258.34905497 ],
    [ 1498679400, 125926053.868564 ],
    [ 1498679520, 190207.599757221 ],
    [ 1498679640, 191997.390291729 ]
  ]
}
]
}

```

## Monitoring service (monitoring.cgi)

**Location:** <https://<hostname>/oess/services/monitoring.cgi>

**Description:** Provides information about the current status of nodes for the use of external monitoring systems.

### Method: get\_node\_status

*RO* *V*

Returns whether the node is currently connected to the OESS OpenFlow controller.

Parameter	Required?	Value type	Description
node	yes	string	Name of the node

### Example request:

```

method: get_node_status
node: sdn-sw.minn.net.internet2.edu

```

### Example response:

```
{
  "results": {
    "status": 1, // whether the node is connected to OESS
    "node": "sdn-sw.minn.net.internet2.edu"
  }
}
```

### Method: get\_mpls\_node\_status

*RO* *V*

Returns whether OESS is connected to the node for purposes of MPLS management.

Parameter	Required?	Value type	Description
node	yes	string	Name of the node

#### Example request:

```
method: get_mpls_node_status
node: sdn-sw.minn.net.internet2.edu
```

#### Example response:

```
{
  "results": {
    "status": 0, // whether OESS is connected to the node
    "node": "sdn-sw.minn.net.internet2.edu"
  }
}
```

#### Method: get\_rules\_on\_node

*RO* *V*

Returns information on the current number of OpenFlow rules on a switch and the configured maximum number of OESS-generated rules allowed for that switch.

Parameter	Required?	Value type	Description
node	yes	string	Name of the node to get information on

#### Example request:

```
method: get_rules_on_node
node: rtsw.chic.net.internet2.edu
```

#### Example response:

```
{
  "results": { // hopefully, these fields are easy to understand
    "maximum_allowed_rules_on_switch": "4000",
    "rules_currently_on_switch": 278,
    "node": "rtsw.chic.net.internet2.edu"
  }
}
```

## Traceroute service (traceroute.cgi)

**Location:** <https://<hostname>/oess/services/traceroute.cgi>

**Description:** interface to run and query OESS's "Trace Circuit Path" functionality, which is like a traceroute, but operates at layer 2 and traces frames through the nodes making up an OpenFlow-based OESS circuit.

The API workflow for a traceroute is:

1. Use `init_circuit_traceroute` to start a traceroute.
2. Periodically poll the status of the traceroute using `get_circuit_traceroute` until it finishes or times out.

## Method: init\_circuit\_traceroute

R W C

Request a traceroute to be performed on a circuit, starting from the specified endpoint.

Parameter	Required?	Value type	Description
circuit_id	yes	integer	ID of the circuit to do the traceroute on
workgroup_id	yes	integer	ID of the workgroup to perform the traceroute as
node	yes	string	Endpoint node to send traceroute frames from
interface	yes	string	Endpoint interface on node to send traceroute frames from

### Example request:

```
method: init_circuit_traceroute
workgroup_id: 491
circuit_id: 306312
node: sdn-sw.miss2.net.internet2.edu
interface: xe-5/0/0.0
```

### Example response:

```
{
  "results": [
    {
      "success": "1" // whether the traceroute was actually started
    }
  ]
}
```

## Method: get\_circuit\_traceroute

R W C

Returns the status of the currently-running (or, if none is running, last-run) traceroute on a circuit.

Parameter	Required?	Value type	Description
circuit_id	yes	integer	ID of the circuit whose traceroute we want to know about
workgroup_id	yes	integer	ID of the workgroup to run as

### Example request:

```
method: get_circuit_traceroute
workgroup_id: 491
circuit_id: 306312
```

### Example response:

```
{
  "results": [ // zero- or one-element array
    {
      "interfaces_traversed": [ // names of interfaces the frames have been traced through, in order
        "et-4/1/0.0",
        "et-4/1/0.1",
        "et-4/1/0.2",
        "et-4/1/0.3"
      ]
    }
  ]
}
```

```

        "et-7/3/0.0",
        "et-7/0/0.0"
    ],
    "end_epoch": null, // when the traceroute completed, timed out, or failed, in seconds
    since the Unix epoch
    "remaining_endpoints": 1, // the number of endpoint nodes yet to be reached
    "nodes_traversed": [ // names of the nodes the frames have been traced through;
    nodes_traversed[i] corresponds with interfaces_traversed[i]
        "rtsw.seat.net.internet2.edu",
        "rtsw.salt.net.internet2.edu",
        "sdn-sw.reno.net.internet2.edu"
    ],
    "ttl": 27, // number of remaining link hops allowed (like an IP packet's TTL)
    "status": "active", // status of the traceroute: {active,Complete,timeout,timed out,
    invalidated}
    "start_epoch": 1498684459 // when the traceroute was started, in seconds since the Unix
    epoch
}
]
}

```

## Workgroup/ACL management service (workgroup\_manage.cgi)

**Location:** [https://<hostname>/oess/services/workgroup\\_manage.cgi](https://<hostname>/oess/services/workgroup_manage.cgi)

**Description:** Provides information about workgroups and interface ACLs, as well as methods to add/modify/remove interface ACLs.

Method: get\_all\_workgroups

*RW* *forall*

Returns the list of all workgroups.

*No parameters.*

**Example request:**

**method:** get\_all\_workgroups

**Example response:**

```
{
    "results": [ // list of workgroups
    {
        "workgroup_id": "121", // numeric ID of workgroup
        "external_id": null, // string identifier available for use by external systems to
        associate something with the workgroup (optional)
        "name": "AutoR&E", // (OEES) name of workgroup
        "type": "normal" // workgroup type: {normal,admin,demo}
    },
    {
        "workgroup_id": "1881",
        "external_id": "I2-S88888",
        "name": "BioRuritania",
        "type": "normal"
    },
}
```

```
{
  "workgroup_id": "42",
  "external_id": null,
  "name": "Demo",
  "type": "demo"
},
{
  "workgroup_id": "101",
  "external_id": null,
  "name": "OEES Admins",
  "type": "admin"
},
[...]
}
```

### Method: get\_acls

**RW** **forall**

Returns a list of interface ACLs.

Parameter	Required?	Value type	Description
interface_id	no	integer	If specified, limits returned ACLs to those for the interface with the given ID
interface_acl_id	no	integer	If specified, only returns the ACL with the given ID

#### Example request:

```
method: get_acls
interface_id: 60551
```

#### Example response:

```
{
  "results": [ // list of ACLs matching parameter criteria
  {
    "interface_name": "xe-4/2/0.0", // name of the interface the ACL applies to
    "vlan_end": "4089", // ending VLAN tag (inclusive) affected by this ACL
    "vlan_start": "1", // starting VLAN tag (inclusive) affected by this ACL
    "owner_workgroup_name": "Performance Assurance", // name of the workgroup that owns the interface
    "workgroup_id": "491", // numeric ID of the workgroup this ACL applies to
    "interface_id": "60551", // numeric ID of the interface the ACL applies to
    "interface_acl_id": "13451", // numeric ID of this ACL
    "workgroup_name": "Performance Assurance", // name of the workgroup this ACL applies to
    "eval_position": "10", // see add_acl method for description
    "owner_workgroup_id": "491", // numeric ID of workgroup that owns the interface
    "notes": null, // any textual notes about this ACL
    "allow_deny": "allow" // whether this ACL permits or denies access to the VLAN tag
  range: {allow,deny}
  },
  {
    "interface_name": "xe-4/2/0.0",
    "vlan_end": "3003",
    "vlan_start": "3003",
    "owner_workgroup_name": "Performance Assurance",
```

```

    "workgroup_id": "337",
    "interface_id": "60551",
    "interface_acl_id": "16441",
    "workgroup_name": "GlobalNOC Server Test Ports",
    "eval_position": "20",
    "owner_workgroup_id": "491",
    "notes": null,
    "allow_deny": "allow"
  },
  {
    "interface_name": "xe-4/2/0.0",
    "vlan_end": "4089",
    "vlan_start": "1",
    "owner_workgroup_name": "Performance Assurance",
    "workgroup_id": "101",
    "interface_id": "60551",
    "interface_acl_id": "28351",
    "workgroup_name": "OESS Admins",
    "eval_position": "30",
    "owner_workgroup_id": "491",
    "notes": null,
    "allow_deny": "allow"
  }
]
}

```

#### Method: add\_acl

**RW IN**

Adds an ACL for a specific interface and a specific workgroup. Note that ACLs for an interface are evaluated in order of increasing **eval\_position**, and evaluation uses a first-match-wins rule. So, if workgroup X has two ACLs on an interface, one with **eval\_position** 20 that allows VLAN tags 1000–1999 and one with **eval\_position** 10 that denies VLAN tags 1500–1509, the workgroup would *not* be allowed to use VLAN tag 1504 on that interface, as the ‘deny’ rule matches first. If the deny rule had **eval\_position** 30 instead, the workgroup *would* be allowed to use VLAN tag 1504.

Parameter	Required?	Value type	Description
interface_id	yes	integer	ID of the interface to add an ACL for
eval_position	no	integer	Position of the new ACL in the order of evaluation. Must not be used by an existing ACL. If not specified, the ACL will be given a position after any existing ACLs for the interface.
workgroup_id	no	integer	ID of the workgroup the ACL applies to; if not specified or set to zero, the ACL applies to all workgroups
entity_id	yes	integer	ID of the Network Entity associated to this VLAN range.
allow_deny	yes	{allow,deny}	Whether the ACL grants or denies access to the VLAN tag range in question
vlan_start	yes	integer	First VLAN tag in the range affected by the ACL
vlan_end	no	integer	Last VLAN tag in the range affected by the ACL; if not specified, defaults to vlan_start

notes	no	string	Short, human-readable note about the ACL
-------	----	--------	--

**Example request:**

```

method: add_acl
interface_id: 60551
eval_position: 15
workgroup_id: 337
entity_id: 23
allow_deny: allow
vlan_start: 504
vlan_end: 521
notes: ACL allowing workgroup 337 onto Demo Network (entity 23) via VLANs 504-521

```

**Example response:**

```
{
  "results": [
    {
      "success": 1, // whether the ACL was actually added
      "interface_acl_id": "32492" // numeric ID of ACL
    }
  ]
}
```

[Method: update\\_acl](#)

**RW IN**

Updates an interface ACL. See the documentation for `add_acl` for details about ACL semantics and user requirements.

Parameter	Required?	Value type	Description
interface_acl_id	yes	integer	ID of the interface ACL to modify
interface_id	yes	integer	ID of the interface the ACL applies to; may not be changed
eval_position	yes	integer	New position of the new ACL in the order of evaluation
workgroup_id	no	integer	ID of the workgroup the ACL applies to; if not specified or set to zero, the ACL applies to all workgroups
entity_id	no	integer	ID of the Network Entity associated to this VLAN range.
allow_deny	yes	{allow,deny}	Whether the ACL grants or denies access to the VLAN tag range in question
vlan_start	yes	integer	First VLAN tag in the range affected by the ACL
vlan_end	no	integer	Last VLAN tag in the range affected by the ACL; if not specified, defaults to <code>vlan_start</code>
notes	no	string	Short, human-readable note about the ACL

**Example request:**

```
method: update_acl  
interface_acl_id: 32492  
interface_id: 60551  
eval_position: 10  
workgroup_id: 337  
allow_deny: allow  
vlan_start: 504  
vlan_end: 599  
notes: Test ACL
```

**Example response:**

```
{  
  "results": [  
    {  
      "success": 1 // whether or not the ACL was actually updated  
    }  
  ]  
}
```

[Method: remove\\_acl](#)

**RW IN**

Removes an existing ACL. User requirements are the same as for add\_acl.

Parameter	Required?	Value type	Description
interface_acl_id	yes	integer	ID of the ACL to remove

**Example request:**

```
method: remove_acl  
interface_acl_id: 32492
```

**Example response:**

```
{  
  "results": [  
    {  
      "success": 1 // whether the ACL was actually removed  
    }  
  ]  
}
```

[Command service \(command.cgi\)](#)

**Location:** <https://<hostname>/oess/services/command.cgi>

**Description:** Provides an interface for executing pre-defined CLI commands via a web GUI.

[Method: get\\_commands](#)

**RO IN**

Returns a list of commands.

Parameter	Required?	Value type	Description
type	no	string	Filters result by command type. Valid types are 'node', 'interface', 'unit', and 'peer'.

**Example request:**

**method:** get\_commands

**Example response:**

```
{
  "results": [
    {
      "command_id": 1, // Database ID of this command
      "name": "version", // Command name displayed to user
      "template": "show version", // Command template executed on device
      "type": "node", // Type of command. Denotes params passed to template
    },
    {
      "command_id": 2,
      "name": "show interface",
      "template": "show interfaces [% interface %].[% unit %]",
      "type": "unit",
    }
  ]
}
```

**Method: run\_command**

**RO IN**

Execute a specified command.

Parameter	Required?	Value type	Description
command_id	yes	integer	Database ID of command to execute.
workgroup_id	yes	integer	Database ID of workgroup to run command as.
node	no	string	Name of node to execute command on
interface	no	string	Name of interface to execute command on
unit	no	string	Unit number to execute command on
peer	no	string	Peer to execute command on

**Example request:**

```
method: run_command
workgroup_id: 1
command_id: 2
node: test.switch.edu
interface: xe-7/0/2
unit: 300
```

**Example response:**

```
{
  "results": [
    "show interfaces xe-7/0/2.300
      Logical interface xe-7/0/2.300 (Index ..."
  ]
}
```

## VRF service (vrf.cgi)

**Location:** <https://<hostname>/oess/services/vrf.cgi>

**Description:** Provides functions relating to the provisioning of VRFs or L3VPNs.

### Method: get\_vrf\_details

*RO C*

Returns the vrf associated with vrf\_id.

Parameter	Required?	Value type	Description
workgroup_id	yes	integer	ID of workgroup to query for
vrf_id	yes	string	Name of node

#### Example request:

```
method: get_vrf_details
workgroup_id: 1
vrf_id: 23
```

#### Example response:

```
{
  "results": [
    {
      "last_modified": "1547049152", // Date last modified
      "last_modified_by": { // User details of modifier
        "email": "user@demo.edu",
        "user_id": "1",
        "last_name": "one",
        "first_name": "user"
      },
      "created": "1547049152", // Date created
      "created_by": { // User details of creator
        "email": "user@demo.edu",
        "user_id": "1",
        "last_name": "one",
        "first_name": "user"
      }
    }
  ],
  "name": "First-VRF", // Name of VRF
  "description": "First-VRF", // Description of VRF
  "vrf_id": "23", // ID of this VRF
  "endpoints": [ // VRF endpoints
```

```
{
  "inner_tag": null, // Inner tag of this endpoint. Defined only in case of QnQ
  "peers": [ // Peerings on this endpoint
    {
      "vrf_ep_peer_id": "23",
      "ip_version": 4,
      "state": "active",
      "peer_ip": "192.168.2.2/31", // Remote peer address
      "peer_asn": "65000",
      "vrf_ep_id": "23",
      "local_ip": "192.168.2.3/31", // Local (OESS) peer address
      "md5_key": "", // BGP Authentication Key
      "operational_state": "up" // State of BGP peering
    }
  ],
  "vrf_endpoint_id": "23",
  "vrf_id": "23",
  "node": { // Details of this endpoint's node
    "longitude": "-1",
    "node_id": "23",
    "latitude": "1",
    "name": "rtsw1.demo.edu"
  },
  "unit": "1001", // Junos specific unit on network device
  "bandwidth": "0",
  "entity": {
    "name": "Demo Network",
    "description": "Demo Network",
    "logo_url": null,
    "entity_id": "23",
    "url": null,
    "parents": Array[1][
      {
        "entity_id": "1",
        "url": null,
        "name": "Root Network",
        "logo_url": null,
        "description": "Root Network"
      }
    ]
  },
  "interface": { // Physical interface of this endpoint
    "cloud_interconnect_id": null,
    "interface_id": "23",
    "name": "et-0/1/5",
    "node": "rtsw1.demo.edu",
    "description": "...",
    "cloud_interconnect_type": null,
    "node_id": "23",
    "operational_state": "up"
  },
  "type": "vrf",
  "tag": "1001", // VLAN or outer STAG of this endpoint
  "cloud_connection_id": null // Cloud connection ID if connected to cloud provider
}
```

```

        },
        ...
    ],
    "state": "active", // State of this VRF
    "local_asn": "55038",
    "workgroup": { // Workgroup owner of this VRF
        "max_circuits": "20",
        "workgroup_id": "23",
        "external_id": "",
        "name": "Demo",
        "type": "normal"
    },
    "prefix_limit": 1000,
    "operational_state": "up",
}
]
}

```

### Method: get\_vrfs

**RO C**

Returns a list of all VRFs that workgroup\_id may view.

Parameter	Required?	Value type	Description
workgroup_id	yes	integer	ID of workgroup to query for

### Example request:

```

method: get_vrfs
workgroup_id: 1

```

### Example response:

```

{
    "results": [
        {
            "last_modified": "1547049152", // Date last modified
            "last_modified_by": { // User details of modifier
                "email": "user@demo.edu",
                "user_id": "1",
                "last_name": "one",
                "first_name": "user"
            },
            "created": "1547049152", // Date created
            "created_by": { // User details of creator
                "email": "user@demo.edu",
                "user_id": "1",
                "last_name": "one",
                "first_name": "user"
            }
        },
        {
            "name": "First-VRF", // Name of VRF
            "description": "First-VRF", // Description of VRF
            "vrf_id": "23", // ID of this VRF
        }
    ]
}

```

```

"endpoints": [ // VRF endpoints
{
  "inner_tag": null, // Inner tag of this endpoint. Defined only in case of QnQ
  "peers": [ // Peerings on this endpoint
    {
      "vrf_ep_peer_id": "23",
      "ip_version": 4,
      "state": "active",
      "peer_ip": "192.168.2.2/31", // Remote peer address
      "peer_asn": "65000",
      "vrf_ep_id": "23",
      "local_ip": "192.168.2.3/31", // Local (OESS) peer address
      "md5_key": "", // BGP Authentication Key
      "operational_state": "up" // State of BGP peering
    }
  ],
  "vrf_endpoint_id": "23",
  "vrf_id": "23",
  "node": { // Details of this endpoint's node
    "longitude": "-1",
    "node_id": "23",
    "latitude": "1",
    "name": "rtsw1.demo.edu"
  },
  "unit": "1001", // Junos specific unit on network device
  "bandwidth": "0",
  "entity": {
    "name": "Demo Network",
    "description": "Demo Network",
    "logo_url": null,
    "entity_id": "23",
    "url": null,
    "parents": Array[1][
      {
        "entity_id": "1",
        "url": null,
        "name": "Root Network",
        "logo_url": null,
        "description": "Root Network"
      }
    ]
  },
  "interface": { // Physical interface of this endpoint
    "cloud_interconnect_id": null,
    "interface_id": "23",
    "name": "et-0/1/5",
    "node": "rtsw1.demo.edu",
    "description": "...",
    "cloud_interconnect_type": null,
    "node_id": "23",
    "operational_state": "up"
  },
  "type": "vrf",
  "tag": "1001", // VLAN or outer STAG of this endpoint
}
]

```

```

    "cloud_connection_id": null // Cloud connection ID if connected to cloud provider
  },
  ...
],
"state": "active", // State of this VRF
"local_asn": "55038",
"workgroup": { // Workgroup owner of this VRF
  "max_circuits": "20",
  "workgroup_id": "23",
  "external_id": "",
  "name": "Demo",
  "type": "normal"
},
"prefix_limit": 1000,
"operational_state": "up",
},
...
]
}

```

### Method: provision

**RWW**

Creates an new vrf. If vrf\_id is specified an edit is performed on that vrf.

Parameter	Required?	Value type	Description
workgroup_id	yes	integer	ID of workgroup to query for
vrf_id	no	string	Name of node
description	Yes	String	Description of VRF
name	Yes	String	Name of VRF
local_asn	Yes	Integer	ASN of local (oess) peer
endpoint	Yes	JSON	JSON object describing desired Endpoint

### Example request:

```

method: provision
description: my first vrf
endpoint: {
  "entity": "Demo Network", // Network Entity to connect to
  "tag": 300, // VLAN to use for connection
  "bandwidth": 0, // Max Bandwidth on Endpoint. 0 is Unlimited
  "peerings": [ // Endpoint peering details
    "asn": 100, // ASN of Peer
    "key": "", // BGP authentication key
    "local_ip": "192.168.2.3/31", // Local (OESSION) peering address
    "peer_ip": "192.168.2.2/31", // Remote peering address
    "version": 4, // IP Version of Peering
  ]
}

```

```
endpoint: { ... } // Multiple Endpoints must be defined  
local_asn: 55038 // Local (OEES) ASN. Use 55038 on al2s.net.internet2.edu  
name: my first vrf  
workgroup_id: 1 // Workgroup used to create VRF. Will become VRF owner
```

**Example response:**

```
{  
  "results": [  
    { "status": "success" }  
  ]  
}
```

## User service (user.cgi)

**Location:** <https://<hostname>/oess/services/user.cgi>

**Description:** Provides functions relating to OEES users.

### Method: get\_current

RO ✓

Returns the details of the currently logged in user.

Parameter	Required?	Value type	Description
-----------	-----------	------------	-------------

**Example request:**

```
method: get_current
```

**Example response:**

```
{  
  "results": [  
    {  
      "user_id": "7",  
      "username": "user",  
      "email": "user@demo.edu",  
      "first_name": "demo",  
      "last_name": "user",  
      "type": "normal", // Indicates if read-only or normal  
      "is_admin": "0", // 1 if user is an admin  
      "workgroups": [ // Workgroups user is in  
        { ... },  
        ...  
      ]  
    }  
  ]  
}
```

## Method: get\_user\_details

RO ADMIN

Returns the details of the specified user.

Parameter	Required?	Value type	Description
user_id	yes	integer	Database ID of user to query

### Example request:

```
method: get_current
user_id: 23
```

### Example response:

```
{
  "results": [
    {
      "user_id": "7",
      "email": "user@demo.edu",
      "first_name": "demo",
      "last_name": "user",
      "type": "normal", // Indicates if read-only or normal
      "is_admin": "0", // 1 if user is an admin
      "workgroups": [ // Workgroups user is in
        { ... },
        ...
      ]
    }
  ]
}
```

## Interface service (interface.cgi)

**Location:** <https://<hostname>/oess/services/interface.cgi>

**Description:** Provides functions relating to OEES network device interfaces.

## Method: get\_available\_vlans

RO C

Returns a list of all VLANs provisionable by workgroup\_id on interface\_id.

Parameter	Required?	Value type	Description
interface_id	Yes	Integer	Database ID of the interface to examine
workgroup_id	Yes	Integer	Database ID of workgroup under which to execute command
vrf_id	No	Integer	If specified will include VLANs already in use by this VRF
circuit_id	No	Integer	If specified will include VLANs already in use by this Circuit

### Example request:

**method:** get\_available\_vlans  
**workgroup\_id:** 1  
**interface\_id:** 71

**Example response:**

```
{
  "results": [
    {
      "available_vlans": [
        100,
        101,
        102,
        ...
      ]
    }
  ]
}
```

**Method: is\_vlan\_available** RO C

Returns 1 if vlan is provisionable by workgroup\_id on interface\_id.

Parameter	Required?	Value type	Description
interface_id	Yes	Integer	Database ID of the interface to examine
workgroup_id	Yes	Integer	Database ID of workgroup under which to execute command
vlan	Yes	Integer	VLAN to check

**Example request:**

**method:** is\_vlan\_available  
**workgroup\_id:** 1  
**interface\_id:** 71  
**vlan:** 101

**Example response:**

```
{
  "results": {
    "allowed": 1
  }
}
```

**Method: get\_workgroup\_interfaces** RO W

Returns a list of interfaces workgroup\_id may provision on.

Parameter	Required?	Value type	Description
workgroup_id	Yes	Integer	Database ID of workgroup under which to execute command

**Example request:**

**method:** get\_workgroup\_interfaces  
**workgroup\_id:** 1

**Example response:**

```
{  
  "results": [  
    {  
      "cloud_interconnect_id": null, // ID indicating physical cloud interconnect  
      "interface_id": "50", // Interface ID  
      "name": "et-2/0/0.0", // Interface name  
      "node": "rtsw.demo.edu", // Network device of interface  
      "description": "BACKBONE ", // Description of interface  
      "cloud_interconnect_type": null, // Type of cloud interconnect  
      "node_id": "23", // ID of network device of interface  
      "acls": [ // List of ACLs indicating provisioning permissions  
        {  
          "workgroup_id": "1",  
          "interface_id": "23",  
          "interface_acl_id": "31",  
          "end": null,  
          "eval_position": "20",  
          "entity_id": null,  
          "allow_deny": "allow",  
          "notes": null,  
          "start": "4070"  
        }  
      ],  
      "operational_state": "up" // State of network interface  
    },  
    ...  
  ]}
```

## Entity service (entity.cgi)

**Location:** <https://<hostname>/oess/services/entity.cgi>

**Description:** Provides functions relating to OESS entities. An entity is essentially a network that may be connected to and is delineated by one or more ACLs. As an example: The entity, University X, may be provisioned to via router.chicago.net on port xe-7/0/1 using VLANs 300-400.

### Method: add\_child\_entity

**RW ADMIN**

Creates a new entity as a child of current\_entity\_id.

Parameter	Required?	Value type	Description
entity_id	Yes	Integer	Identifier of the entity to query
user_id	Yes	Integer	Identifier of first user. All entities must have at least one user.
name	Yes	String	Name of Entity
description	No	String	Description of Entity
url	No	String	URL to entity's external site
logo_url	No	String	URL to entity's logo

#### Example request:

```
method: add_child_entity
entity_id: 1
user_id: 1
name: Simple Entity
```

#### Example response:

```
{
  "results": [
    {
      "success": 1,
      "child_entity_id": 5
    }
  ]
}
```

### Method: add\_user

**RW C**

Associates an existing user with an entity.

Parameter	Required?	Value type	Description
entity_id	Yes	Integer	Identifier of the entity to modify
user_id	Yes	Integer	Identifier of user to add

#### Example request:

```
method: add_user
entity_id: 1
```

**user\_id:** 1

**Example response:**

```
{  
  "results": [  
    {  
      "success": 1  
    }  
  ]  
}
```

**Method: get\_entities**

*RO* *V*

Gets a list of entities matching the requested criteria.

Parameter	Required?	Value type	Description
workgroup_id	Yes	Integer	Identifier of workgroup executing this command
name	No	String	Entity name to filter by; May be a partial match.

**Example request:**

```
method: get_entities  
workgroup_id: 1  
name: indi
```

**Example response:**

```
{  
  "results": [  
    {  
      "contacts": [ ... ],  
      "name": "Indiana University",  
      "children": [ ... ],  
      "description": "",  
      "logo_url": "https://localhost/icon.png",  
      "interfaces": [ ... ],  
      "entity_id": 11,  
      "url": "https://localhost/",  
      "parents": [ ... ]  
    },  
    ...  
  ]  
}
```

**Method: get\_entity**

*RO* *V*

Gets an entity by entity\_id.

Parameter	Required?	Value type	Description
entity_id	No	Integer	Identifier of the entity to filter by
workgroup_id	No	Integer	Identifier of Workgroup to filter Entities by
vrf_id	No	Integer	Identifier of VRF to filter Entities by
circuit_id	No	Integer	Identifier of Circuit to filter Entities by

**Example request:**

```
method: get_entity
entity_id: 1
workgroup_id: 1
```

**Example response:**

```
{
  "results": [
    {
      "contacts": [ ... ],
      "name": "Indiana University",
      "children": [ ... ],
      "description": "",
      "logo_url": "https://localhost/icon.png",
      "interfaces": [ ... ],
      "allowed_vlans": [ ... ],
      "entity_id": 11,
      "url": "https://localhost/",
      "parents": [ ... ]
    },
    ...
  ]
}
```

**Method: get\_entity\_children**

*RO* *forall*

Gets all child entities of the specified entity.

Parameter	Required?	Value type	Description
entity_id	Yes	Integer	Identifier of the parent entity to query

**Example request:**

```
method: get_entity_children
entity_id: 1
```

**Example response:**

```
{
  "results": [
    {
      "contacts": [ ... ],
      "name": "Indiana University",
      "children": [ ... ],
      "description": "",
      "logo_url": "https://localhost/icon.png",
      "interfaces": [ ... ],
      "entity_id": 11,
      "url": "https://localhost/",
      "parents": [ ... ]
    },
    ...
  ]
}
```

```
    ]  
}
```

### Method: get\_entity\_interfaces

RO

Gets all interfaces of the specified entity.

Parameter	Required?	Value type	Description
entity_id	Yes	Integer	Identifier of the entity to query

#### Example request:

```
method: get_entity_interfaces  
entity_id: 1
```

#### Example response:

```
{  
  "results": [  
    {  
      "cloud_interconnect_id": null,  
      "cloud_interconnect_type": null,  
      "interface_id": 23,  
      "name": "xe-7/0/1",  
      "node_id": "23",  
      "node": "router.demo.net",  
      "description": "",  
      "operational_state": "up",  
      "acls": [ ... ]  
    }  
  ]  
}
```

### Method: get\_root\_entities

RO

Gets all direct descendants of the root entity.

Parameter	Required?	Value type	Description
-----------	-----------	------------	-------------

#### Example request:

```
method: get_root_entities
```

#### Example response:

```
{  
  "results": [  
    {  
      "contacts": [  
        ...  
      ],  
      "name": "Connectors",  
      "children": [  
        ...  
      ],  
      "id": "1"  
    }  
  ]  
}
```

```

    "description": "",
    "logo_url": "https://localhost/icon.png",
    "interfaces": [
        ...
    ],
    "entity_id": 11,
    "url": "https://localhost/",
    "parents": [
        ...
    ]
}
]
}

```

### Method: get\_valid\_users RO

Gets all user Identifiers of the specified entity.

Parameter	Required?	Value type	Description
entity_id	Yes	Integer	Identifier of the entity to query

### Example request:

```

method: get_valid_users
entity_id: 1

```

### Example response:

```
{
    "results": [
        "12",
        "23"
    ]
}
```

### Method: remove\_user RW

Removes user from the specified entity.

Parameter	Required?	Value type	Description
entity_id	Yes	Integer	Identifier of the entity to modify
workgroup_id	Yes	Integer	Identifier of user to remove

### Example request:

```

method: remove_user
entity_id: 1
user_id: 1

```

### Example response:

```
{
    "results": [
        {
    }
```

```
        "success": 1
    }
]
}
```

### Method: update\_entity

RW W

Updates the specified entity using the provided params.

Parameter	Required?	Value type	Description
entity_id	Yes	Integer	Identifier of the entity to query
description	No	String	Desired description of entity
name	No	String	Desired name of entity
url	No	String	Desired link to entity organization's web presence
logo_url	No	String	Desired link to source image of this entity

#### Example request:

```
method: update_entity
entity_id: 1
description: "A new description"
```

#### Example response:

```
{
  "results": [
    {
      "success": 1
    }
  ]
}
```